VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

VÝVOJ WEBOVÝCH APLIKACÍ POMOCÍ JAZYKA XUL

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

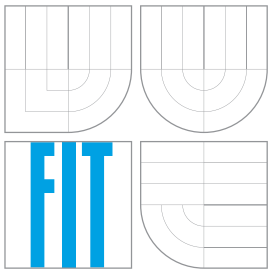
AUTOR PRÁCE
AUTHOR

ONDŘEJ RAUL

BRNO 2011

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY

DEPARTMENT OF INFORMATION SYSTEMS

VÝVOJ WEBOVÝCH APLIKACÍ POMOCÍ JAZYKA XUL

WEB-APPLICATION DEVELOPMENT IN XML USER INTERFACE LANGUAGE

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — ONDŘEJ RAUL

AUTHOR

VEDOUCÍ PRÁCE — Mgr. MAREK RYCHLÝ, Ph.D.

SUPERVISOR

BRNO 2011

Abstrakt

Cílem práce je představit jazyk XML User Interface Language (XUL) a srovnat postup tvorby aplikací pomocí XUL s existujícími rámci pro vývoj webových aplikací. Dále se práce zabývá návrhem internetového obchodu a jeho implementací pomocí jazyka XUL. Rozhraní aplikace pro nakupujícího bude implementované v prostém HTML. Nakonec srovnává uživatelské rozhraní v XUL s rozhraním v HTML a hodnotí vhodnost jazyka XUL pro tvorbu internetových aplikací.

Abstract

Aim of the thesis is to introduce the XML User Interface Language (XUL) and compare the process of building web applications in XUL with the existing frameworks for developing web applications. Moreover the thesis deals with a design of an e-shop and its implementation in XUL language. The interface for customers will be implemented in HTML. In the end the thesis compares the user interface in XUL with the user interface in HTML and evaluates the suitability of XUL language for building web applications.

Klíčová slova

XUL, (X)HTML, XML, DHTML, PHP, MySQL, JavaScript, webové aplikace, rámce pro vývoj webových aplikací

Keywords

XUL, (X)HTML, XML, DHTML, PHP, MySQL, JavaScript, web applications, web application frameworks

Citace

Ondřej Raul: Vývoj webových aplikací pomocí jazyka XUL, bakalářská práce, Brno, FIT VUT v Brně, 2011

Vývoj webových aplikací pomocí jazyka XUL

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Mgr. Marka Rychlého, Ph.D.

.....

Ondřej Raul

15. května 2011

Poděkování

Na tomto místě bych rád poděkoval vedoucímu této práce, panu Mgr. Markovi Rychlému, Ph.D. za odborné konzultace, vstřícnost a trpělivost.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Cílem bakalářské práce je představit jazyk pro tvorbu grafického uživatelského rozhraní XML User Interface Language, zkráceně XUL[1], popsat postup tvorby aplikací pomocí jazyka XUL a tento postup srovnat s existujícími rámci pro vývoj internetových aplikací. Dále se práce věnuje návrhu internetového obchodu, kde je část pro nakupující uživatele implementována pomocí prostého HTML a administrátorská část pro prodejce pomocí XUL.

Práce je rozdělena do několika kapitol. Druhá kapitola uvádí jazyky a technologie použité při tvorbě internetového obchodu. Třetí kapitola představuje jazyk XUL spolu s postupem tvorby aplikací pomocí tohoto jazyka. Kapitola dále doporučuje některé nástroje pro podporu programování v XUL. Na závěr srovnává XUL s několika dalšími rámci pro vývoj webových aplikací. Čtvrtá kapitola se věnuje návrhu aplikace a pátá kapitola její implementaci. Tato kapitola je rozdělena na dvě části, část pro nakupující a část pro prodejce. Obě části se zaměřují na hlavní rysy zvoleného přístupu implementace. V poslední, šesté kapitole, se nachází závěrečné zhodnocení práce spolu s možnostmi jejího rozšíření.

[1] Vyslovuje se „zůl“ podle jména ducha Sumérského boha z filmu Krotitelé duchů [48].

Kapitola 2

Použité technologie

V této kapitole jsou ve zkratce představeny a popsány všechny technologie a jazyky, které byly použity při implementaci aplikace internetového obchodu.

2.1 XML

Extensible Markup Language je značkovací jazyk vytvořený zjednodušením jazyka SGML (*Standard Generalized Markup Language*). Mezi nejvýraznější změny patří odstranění nepárových značek a zpřísnění syntaxe (tam, kde bylo u SGML více možností zápisu, je v XML jen jedna). Díky těmto změnám se stal jazyk XML hodně rozšířeným a využívaným. Používá se k ukládání dat, ke komunikaci, v databázových systémech a v mnoha dalších aplikacích. [36]

2.2 Dynamické HTML

Dynamické HTML neboli DHTML je termín označující soubor technologií používaných ke tvorbě dynamických internetových stránek, tedy stránek, které jsou interaktivní a jistým způsobem mění svůj obsah. Mezi tyto technologie patří značkovací jazyk HTML (*Hypertext Markup Language*) nebo XHTML, skriptovací jazyk JavaScript, jazyk pro definici vzhledu CSS a DOM. Tímto spojením lze dosáhnout různých efektů, jako je například animovaný text či obrázek, vyjíždějící menu, změna vzhledu apod. [6]

Nyní si jednotlivé části představíme podrobněji.

2.2.1 XHTML

Extensible Hypertext Markup Language je značkovací jazyk pro tvorbu hypertextových dokumentů vytvořený z jazyka HTML. Byl upraven tak, aby odpovídal standardu XML. Většina značek byla zachována a na první pohled není patrné, zda-li se jedná o jazyk HTML či XHTML. Došlo pouze k drobným změnám, mezi které patří například absence některých značek pro definici vzhledu, přidaní XML hlavičky apod. Výhodou XHTML je, že se jedná o validní XML dokument a můžeme tedy použít všechny nástroje pro zpracování XML dokumentů (například *XSL Transformaci*[1]). Nejsme tak nuceni využívat pouze nástroje pracující s HTML dokumenty. [36]

Kompletní specifikaci, rozdíly oproti HTML a další informace lze najít v [23], kde je popsána verze XHTML 1.0, verze XHTML 1.1 je pak v [24].

V ukázkové aplikaci je použita verze XHTML 1.0 Strict.

2.2.2 CSS

Kaskádové styly (*Cascading Style Sheets*) jsou jazyk sloužící k popisu vzhledu dokumentů napsaných v některém ze značkovacích jazyků (HTML, XHTML, ...). Dalo by se říci, že jazyk CSS je nadstavbou pro tyto jazyky. Umožňuje definovat způsob zobrazení těchto dokumentů, aniž by jakkoliv ovlivnil jejich obsah, strukturu nebo sémantiku.

Definice vzhledu je oddělena od samotného textu, což přináší své výhody. Máme možnost definovat různý vzhled pro různá zobrazovací zařízení (monitor, tiskárna, ...), dokument se stane přehlednějším, můžeme mít několik různých vzhledů pro jeden dokument a ty libovolně měnit (například v závislosti na dni v týdnu) apod. [46]

Existuje několik verzí jazyka CSS, verze 1, 2 a 3. Každá staví na té předchozí a typicky se přidávají nové vlastnosti. První verze CSS vznikla v roce 1996. V současnosti se pracuje na CSS3. Tato verze se začala vyvíjet v roce 2005. V [2] lze najít kompletní specifikace jednotlivých verzí. [3]

2.2.3 JavaScript

Je programovací jazyk používaný především k tvorbě programů běžících na straně klienta, a to nejčastěji ve webovém prohlížeči. Jedná se o interpretovaný jazyk, to znamená, že

[1] Extensible Stylesheet Language Transformations. Více viz http://cs.wikipedia.org/wiki/XSLT.

program je přímo vykonáván, aniž by byl překládán do strojového kódu (interpretem je obvykle výše zmíněný webový prohlížeč). Syntaxí se podobá jazyku Java, případně C/C++. Jinak ale nemá s jazykem Java nic společného. JavaScript umožňuje objektově orientované programování, avšak nejedná se o jazyk výhradně objektově orientovaný. [36]

Většina dnešních webových prohlížečů, jako je např. Mozilla nebo Internet Explorer, obsahuje verzi tohoto jazyka, tzv. klientský JavaScript, což je pojem označující integraci JavaScriptu do webového prohlížeče. Problémem je, že verze v jednotlivých prohlížečích se od sebe mohou mírně odlišovat a to pochopitelně ztěžuje programátorům práci. [40]

JavaScript se uplatňuje především ve webových prezentacích jako nástroj pro zajištění větší interaktivity stránek s uživatelem. Lze takto například ověřovat údaje ve formuláři předtím, než se odešlou na server, což bylo podstatně důležitější dříve, kdy rychlosti připojení k internetu nebyly tak vysoké jako dnes. Odeslat data na server, tam je kontrolovat a posílat uživateli zpět s chybovým hlášením, byl velmi zdlouhavý proces. Mezi další možnosti využití patří mimo jiné také spolupráce s AJAXem, který bude představen později. [49]

2.2.4 DOM

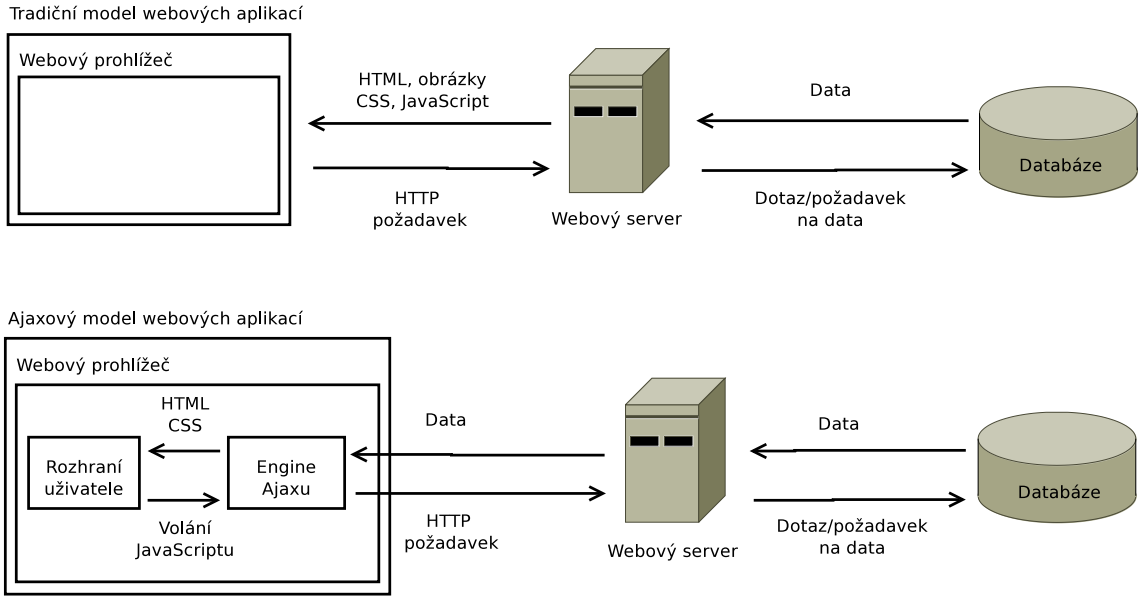
Zkratka DOM pochází z anglického *Document Object Model* - objektový model dokumentu. Jedná se o API (*Application Programming Interface*), pomocí kterého můžeme přistupovat ke struktuře dokumentu. Lze tak zjišťovat vlastnosti jednotlivých prvků, měnit je, přidávat nové nebo nějaké odstranit. DOM nám umožňuje přistupovat k dokumentu jako ke stromu. Hlavní výhoda spočívá ve schopnosti s dokumentem manipulovat bez potřeby znovu se spojit se serverem. Tuto technologii používají především technologie pracující na straně klienta. V [4] lze najít více informací o jednotlivých verzích. [45]

2.3 AJAX

Asynchronous JavaScript and XML je spíše technika než technologie jak vytvářet webové aplikace. Hlavním cílem je minimalizace přenosu dat. Měla by se přenášet jen ta data, která je v daném okamžiku potřeba přenést. Nač opětovně načítat celou stránku, když stačí obnovit jen její část. Sníží se tím nároky na přenos dat na server a také se zrychlí interakce s uživatelem, který nebude muset čekat, než se vše znovu přenese. Navíc bude

moci dále pracovat, zatímco bude probíhat přenos. Technologie, které AJAX využívá, jsou HTML/XHTML, CSS, DOM, XML, JavaScript a objekt *XMLHttpRequest* (XHR). [32]

XHR je objekt umožňující asynchronní komunikaci se serverem pomocí HTTP protokolu. AJAX pomocí něj vytvoří spojení a pošle data na server. Nejčastěji se pro předávání dat používá formát XML. Srovnání tradičního modelu webových aplikací s modelem AJAXu můžeme vidět na obrázku 2.1. [43]



Obrázek 2.1: Srovnání tradičního modelu webových aplikací a Ajaxu [32].

AJAX je tedy přístup, jak může klient asynchronně komunikovat se serverem (v aplikaci e-shopu budeme takto komunikovat se serverem, na kterém poběží PHP). Lze použít všude tam, kde není potřeba obnovovat naráz celou stránku a kde potřebujeme zvýšit interakci s uživatelem. Získáme tak mnohem větší dynamičnost, zmenší se velikost přenášených dat a s tím souvisí zkrácení doby pro odpověď serveru. [32]

S touto technikou se v praxi setkávají například uživatelé Gmailu[2] nebo Google Maps[3]. Úplnou specifikaci lze najít v [26].

[2]E-mailová služba poskytovaná společností Google. Více viz http://mail.google.com.

[3]Internetová mapa poskytovaná společností Google. Více viz http://maps.google.com.

2.4 PHP

Je zkratka pro *PHP: Hypertext Preprocesor*. Jedná se o interpretovaný skriptovací jazyk pracující na straně serveru. PHP bylo navrženo pro práci na webu, což je jeho hlavní úloha. Řada autorů se shodla na tom, že je to jazyk jednoduchý k naučení, robustní a velmi dobře použitelný právě k webovému programování. Syntaxí se podobá mimo jiné i JavaScriptu a neobsahuje žádné složité konstrukce. [37]

Mezi výhody PHP patří velký seznam funkcí usnadňujících programátorovi práci (obsahuje funkce pro práci se soubory, grafikou, databázemi, e-maily, ...), dále rozšíření a podpora, open source licence, podpora objektově orientovaného programování a další. [37]

Procesor jazyka PHP je součástí HTTP serveru, který zpracovává požadavky od klienta. HTTP server přijme požadavek, vyhodnotí jej, a pokud se jedná o PHP soubor, nebo obsahuje PHP požadavek, předá jej PHP procesoru. Ten požadavek vyhodnotí a vygeneruje výstupní dokument, který se odešle klientovi (nejčastěji ve formátu HTML jako běžná stránka). Dnešní HTML stránky běžně obsahují kousky kódu v PHP. Uživatel to ani nepozná, protože PHP příkazy se provedou na serveru a výsledek je zobrazen ve formátu HTML. [35]

Více informací o tomto jazyku, zdrojové kódy ke stažení a další informace lze najít na oficiálních stránkách [14].

2.5 MySQL

MySQL je multiplatformní relační databázový systém, který je dnes hojně používaný především pro svůj výkon, snadnou implementovatelnost, jednoduchou správu a také proto, že jde o volně šiřitelný software[4]. Je podporován takřka všemi webovými hostingy a dobře spolupracuje s PHP. Komunikace s databází probíhá pomocí jazyka SQL (*Structured Query Language*), což je jazyk pro manipulaci s daty v relačních databázích. Na oficiálních stránkách [13] lze najít informace o MySQL, software ke stažení a další údaje. [12]

Pro administraci MySQL databází lze použít nástroj phpMyAdmin, který umožňuje například snadnou tvorbu tabulek, vkládání dat a další.

[4] Pro nekomerční použití. Dostupný pod GPL Licencí - http://www.gnu.org/licenses/gpl.html.

2.6 XUL

Jazyku XUL se podrobněji věnuje kapitola 3. Zde je zmíněn pro kompletnost seznamu použitých technologií.

Kapitola 3

XUL

Následující kapitola si klade za cíl představit jazyk XUL a další technologie, které se ve spojení s XUL používají ke tvorbě aplikací.

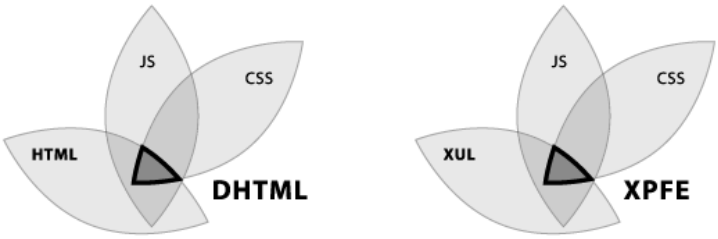
3.1 Představení

XML User Interface Language je, jak již název napovídá, jazyk založený na XML sloužící pro popis uživatelského rozhraní. Byl vyvinut organizací Mozilla Foundation s cílem vytvořit pro multiplatformní aplikace grafické rozhraní, které by bylo lehce modifikovatelné. [34]

Pro webové vývojáře je snadné tento jazyk používat, protože je velmi podobný jazyku HTML. Spolu s XUL se ke tvorbě aplikací používají další webové standardy a technologie, jako například XML, CSS, JavaScript, DOM, XBL, RDF a další. Spojení XUL s kaskádovými styly a skriptovacím jazykem JavaScript se nazývá *XPFE Framework*. Tento framework je podobný jako DHTML, jak můžeme vidět na obrázku 3.1, ve kterém je jazyk XUL nahrazen jazykem HTML. [33]

Vykreslování uživatelského rozhraní v XUL je zajištěno pomocí rendrovacího jádra *Gecko*, které slouží mimo jiné také k vykreslování webových stránek. Pro zobrazení uživatelského rozhraní v XUL je nutné použít nějakou implementaci tohoto jádra, například prohlížeč Firefox či XULRunner. [27, 8]

XULRunner je běhové prostředí pro XUL aplikace. Jeho myšlenka vychází z *Gecko Runtime Environment* (GRE). Více se můžete dozvědět v [30].



Obrázek 3.1: Srovnání DHTML a XPFE [33].

3.1.1 XBL

XML Binding Language (někdy nazývaný také *Extensible Bindings Language*) je značkovací jazyk založený na XML pro popis vazeb, kterými mohou být připojeny elementy v jiných dokumentech. Takto připojený element se nazývá *vázaný element* a získává nové chování definované vazbou. [10]

Existují dvě verze, XBL 1.0 a XBL 2.0. První verze je specifikovaná v [22]. Problémem je, že není standardizovaná a její aktuální implementace v Mozille se od specifikace liší. Tyto problémy se snaží odstranit verze 2.0, kterou vyvíjí *World Wide Web Consortium* (W3C). V [25] je dostupný dokument *W3C Candidate Recommendation* pro XBL 2.0. [10]

V XUL můžeme definovat prvky uživatelského rozhraní, pomocí stylů měnit jejich vzhled a pomocí XBL měnit chování. Můžeme si tedy vytvářet vlastní widgety[1], které budou znovupoužitelné i v jiných projektech. [33]

XBL dokument se ukládá s příponou `.xml` a je to validní XML dokument. Kořenovým elementem je `<bindings>`, který obsahuje libovolný počet vazeb `<binding>`. Každá vazba `<binding>` může dále obsahovat `<content>`, `<implementation>` a `<handlers>`. Všechny prvky uvnitř obsahu tagu `<content>` budou tvořit tělo, `<implementation>` chování, `<handlers>` obsluhu událostí. Jednotlivé tagy mohou ještě obsahovat další atributy (např. `id`). Kompletní přehled syntaxe XBL souboru lze najít v [34, 22]. Příklad, jak může takový XBL soubor vypadat, můžeme vidět na obrázku 3.2. [34]

Pomocí CSS lze navázat vazbu na konkrétní prvek, slouží k tomu atribut `-moz-binding`. Použití CSS ukazuje následující názorný příklad, na kterém lze vidět tvorbu vlastní jed-

[1] Prvek grafického uživatelského rozhraní (GUI).

```
<?xml version="1.0"?>
<bindings xmlns="http://www.mozilla.org/xbl">
  <binding id="vazba1">
    <content>
      <!-- obsah -->
    </content>
    <implementation>
      <!-- chování -->
    </implementation>
    <handlers>
      <!-- obsluha událostí -->
    </handlers>
  </binding>
  <binding id="vazba2">
    <!-- další -->
  </binding>
</bindings>
```

Obrázek 3.2: Syntax XBL souboru.

noduché komponenty skládající se ze dvou tlačítek. Její zdrojový kód je znázorněn na obrázku 3.3, náhled pak na obrázku 3.4. [10]

3.1.2 RDF

Resource Description Framework, česky rámec pro popis zdrojů, slouží pro zápis metadat. Je standardizovaný konsorciem W3C a využívá jazyk XML. Informace se zapisují ve formě trojic (subjekt - predikát - objekt): zdroj má hodnotu určité vlastnosti. Tedy příklad „obloha má modrou barvu" by se pomocí RDF zapsal ve tvaru: subjekt - „obloha", predikát - „má barvu", objekt - „modrá". [18]

Na příkladu[2] znázorněném na obrázku 3.5 je uvedena syntax jednoduchého `.rdf` souboru definujícího, že autorem stránky „http://www.prihoda.cz/"je „Pavel Příhoda".

Templates

XUL nabízí způsob, jak můžeme načítat data z `.rdf` souborů a manipulovat s nimi. Následující příklad na obrázku 3.6 ukazuje, jak načíst RDF data a vytvořit z nich XUL strom.

Další informace o RDF syntaxi lze najít ve W3C specifikaci [16], o XUL templates pak v [38].

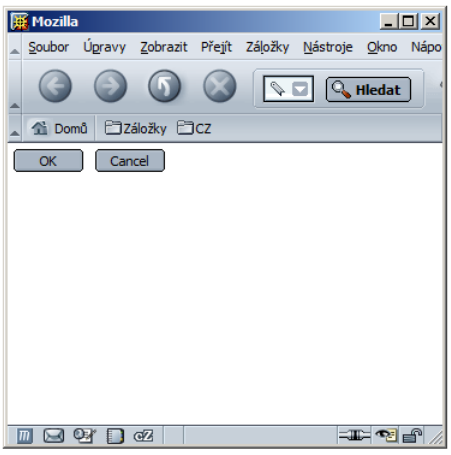
[2] Informace o použitém jmenném prostoru lze najít na http://dublincore.org/documents/1999/07/02/dces/.

```
<!-- soubor vlastnikomponenta.xul -->
<?xml version="1.0"?>
<?xml-stylesheet href="vlastnikomponenta.css" type="text/css"?>
<window
  xmlns=
    "http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">
  <box class="okcanceltlacitka"/>
</window>

<!-- soubor vlastnikomponenta.css -->
box.okcancelbuttons {
  -moz-binding: url('vlastnikomponenta.xml#okcanceltlacitka');
}

<!-- soubor vlastnikomponenta.xml -->
<?xml version="1.0"?>
<bindings xmlns="http://www.mozilla.org/xbl"
  xmlns:xul=
    "http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">
  <binding id="okcanceltlacitka">
    <content>
      <xul:button label="OK"/>
      <xul:button label="Cancel"/>
    </content>
  </binding>
</bindings>
```

Obrázek 3.3: Zdrojový kód vlastní komponenty.



Obrázek 3.4: Screenshot nově vytvořené komponenty.

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/">
    <rdf:Description rdf:about="http://www.prihoda.cz/"
      dc:creator="Pavel Příhoda"/>
</rdf:RDF>
```

Obrázek 3.5: Ukázka zkráceného `.rdf` souboru dostupného v [15].

3.2 Postup tvorby aplikací

Pomocí XUL lze tvořit několik typů aplikací [38]:

- **Firefox extensions** - rozšíření pro Firefox,
- **XUL package**,
- **standalone XULRunner applications** - samostatné aplikace využívající pro svůj běh XULRunner,
- **remote XUL applications** - aplikace běžící vzdáleně na serveru.

My se zde budeme zabývat posledními dvěma odrážkami. Postup vývoje je u obou obdobný.

Při tvorbě aplikací (desktopových), které se budou spouštět pomocí prostředí XULRunner, je nutné dodržet strukturu adresářů a souborů znázorněnou na obrázku 3.7. Důležité soubory jsou:

- `application.ini`,
- `main.xul`,
- `prefs.js`,
- `chrome.manifest`.

Soubor `main.js` není nezbytný. Obsahuje kód v jazyce JavaScript, který je možný umístit přímo do souboru `main.xul`. Stejně tak lze ještě vytvořit další soubor s příponou `.css`, kam je možné umístit definice stylů.

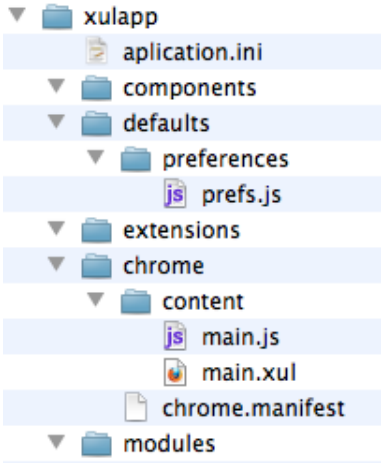
```
<!-- soubor zvirata.rdf -->
<?xml version="1.0"?>
<RDF:RDF xmlns:RDF="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:ANIMALS="http://www.nejaka-zoo.cz/rdf#">

  <RDF:Description RDF:about="http://www.nejaka-zoo.cz/ptaci"
     ANIMALS:jmeno="Ptaci"/>
    <RDF:Description RDF:about="http://www.nejaka-zoo.cz/ptaci/emu"
       ANIMALS:jmeno="Emu" ANIMALS:pocet="3"/>
    <RDF:Description RDF:about="http://www.nejaka-zoo.cz/ptaci/pav"
       ANIMALS:jmeno="pav" ANIMALS:pocet="5"/>

  <RDF:Seq RDF:about="http://www.nejaka-zoo.cz/zvirata">
    <RDF:li>
      <RDF:Seq RDF:about="http://www.nejaka-zoo.cz/ptaci">
        <RDF:li RDF:resource="http://www.nejaka-zoo.cz/ptaci/emu"/>
        <RDF:li RDF:resource="http://www.nejaka-zoo.cz/ptaci/pav"/>
      </RDF:Seq>
    </RDF:li>
  </RDF:Seq>
</RDF:RDF>

<!-- XUL soubor -->
<tree datasources="zvirata.rdf" flex="1"
      ref="http://www.nejaka-zoo.cz/zvirata">
  <treecols>
    <treecol label="Jméno" primary="true" flex="1"/>
    <treecol label="Počet" flex="1"/>
  </treecols>
  <template>
    <rule>
      <treechildren>
       <treeitem uri="rdf:*">
         <treerow>
           <treecell
               label="rdf:http://www.nejaka-zoo.cz/rdf#jmeno"/>
           <treecell
               label="rdf:http://www.nejaka-zoo.cz/rdf#pocet"/>
         </treerow>
       </treeitem>
      </treechildren>
    </rule>
  </template>
</tree>
```

Obrázek 3.6: Upravený příklad demonstrující vytvoření XUL stromu z `.rdf` souboru. Originál dostupný v [19].



Obrázek 3.7: Adresářová struktura XUL aplikace.

Soubor application.ini

Obsahuje požadovanou verzi jádra Gecko, popř. další informace jako jméno aplikace, verzi a další:

```
[App]
Vendor=XULTest
Name=TestApp
Version=1.0
BuildID=20100901
ID=xulapp@xultest.org

[Gecko]
MinVersion=1.8
MaxVersion=2.*
```

Soubor main.xul

Hlavní soubor s uživatelským rozhraním.

Soubor prefs.js

Pomocí něj spouští XULRunner aplikaci. Obsahuje URI hlavního souboru s rozhraním:

```
pref("toolkit.defaultChromeURI",
     "chrome://xulapp/content/main.xul");
```

Soubor chrome.manifest

Zde je určen hlavní adresář aplikace, ve kterém je hlavní soubor s rozhraním, který jsme viděli v předchozím příkladu:

```
content xulapp content/
```

Při tvorbě aplikací běžících vzdáleně na serveru se nemusíme adresářovou strukturou příliš zabývat. Adresáře si vytvoříme dle naší potřeby a stejně tak i soubory. Je nutné jen správně nastavit prostředí na serveru, více v příloze práce.

Nyní se podíváme na několik důležitých věcí, na které nesmíme zapomenout. Každý `.xul` soubor musí začínat XML deklarací:

```
<?xml version="1.0"?>
```

Dále je potřeba určit jmenný prostor (angl. *namespace*). Ten se definuje jako atribut kořenového elementu:

```
<window
  xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">
</window>
```

V korektním XML dokumentu by neměla chybět také document type (*DOCTYPE*) deklarace:

```
<!DOCTYPE window>
```

Budeme-li potřebovat komentáře, značí se stejně jako v HTML:

```
<!-- zde je komentář -->
```

3.2.1 XUL elementy a vlastnosti

Okno aplikace

Kromě klasického kořenového elementu `<window>` máme na výběr několik dalších:

- `<page>` - používá se pro dokumenty, které se vkládají do jiných stránek;
- `<dialog>` - dialogové okno;
- `<overlay>` - speciální typ pro definici vkládaných částí, jako je například menu. Více se jím budeme zabývat později;

- `<wizard>` - okno usnadňující tvorbu průvodců („*step-by-step*”).

Všechny lze doplnit o atributy, více v [1].

Layout

Základní mechanismus pro rozmístění prvků je tzv. box model. S jeho pomocí můžeme rozdělit okno na několik podoken, „boxů”. Rozdělení by se dalo přirovnat k tabulce, kde každý box reprezentuje jednu buňku v tabulce. Takto můžeme okno rozdělit pomocí elementů `<box>`, `<vbox>` a `<hbox>`. Všechny tři se chovají stejně s tím rozdílem, že prvky uvnitř elementu `<box>` a `<hbox>` jsou řazeny horizontálně vedle sebe, u elementu `<vbox>` se pak řadí vertikálně pod sebe. Umístíme-li například dvě tlačítka vertikálního boxu (`vbox`), budou pod sebou. [33]

Dalším elementem pro určení layoutu je `<spacer>`. Ten zabírá místo, ale nic nezobrazuje. Potřebujeme-li mezi dvěma prvky volné místo, poslouží nám právě `<spacer>`.

Všechny výše zmíněné elementy je vhodné doplnit o atribut `flex`, určující jak bude daný prvek flexibilní. Jinými slovy, jak bude ochotný se roztahovat, pokud vedle něj bude volné místo. Umístíme-li dvě stejná tlačítka vedle sebe na stránku a prvnímu přiřadíme atribut `flex` roven jedné a druhému roven devíti, pak se druhé tlačítko roztáhne přes 90% možné plochy, první jenom přes 10%. [1]

Kompletní seznam všech XUL elementů, jejich atributů, metod, ukázek použití a dalších informací lze najít v [1].

Prvky GUI

V XUL máme možnost vybrat si z velkého množství nejrůznějších prvků, od běžných tlačítek, popisků, vstupních polí, přes datepicker, timepicker nebo colorpicker až po menu, listbox či strom. Opravdu máme z čeho vybírat. Výrazné ulehčení práce nabízí především prvky jako je zmíněné menu. Není potřeba zanořovat seznamy jako v HTML, stylovat je ani přidávat JavaScriptové funkce.

Dalším užitečným prvkem, který zde byl už zmíněn, je strom. Umožňuje nám přehledně uspořádat data do sloupců, vytvářet hierarchii. Máme možnost skrývat sloupce, které nás nezajímají a další. Ostatní GUI elementy jsou popsány v [1].

Vkládání HTML

Přidáním atributu označujícího (X)HTML jmenný prostor můžeme vkládat do XUL kód HTML:

```
<window xmlns:html="http://www.w3.org/1999/xhtml"
  xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">

<html:h1>Nadpis první úrovně.</html:h1>
```

Overlays

Při tvorbě aplikací se často setkáváme se situací, kdy potřebujeme stejnou část kódu popisující často nějaký prvek, například menu, vložit do více stránek. Je výhodnější mít menu definované na jednom místě, v jednom souboru, než kopírovat kód do každého souboru zvlášť. K tomu nám slouží právě overlays. V souboru `menu_overlay.xul` si definujeme menu, které pak můžeme libovolně vkládat do dalších souborů (3.8). [34]

```
<!-- soubor menu_overlay.xul -->
<?xml version="1.0"?>
<overlay
  xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.
     xul">
<menu id="hlavni-menu-overlay">
  <menupopup id="hlavni-menu">
    <menuitem id="menuitem-soubor" label="Soubor"/>
    <menuitem id="menuitem-nastaveni" label="Nastavení"/>
    <menuitem id="menuitem-napoveda" label="Nápověda"/>
  </menupopup>
</menu>
</overlay>

<!-- soubor main.xul -->
<?xml version="1.0"?>
<!-- specifikace overlay souboru -->
<?xul-overlay href="menu_overlay.xul"?>
<window
  xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.
     xul">
  <vbox>
        <!-- vložené menu -->
        <menu id="hlavni-menu-overlay"/>
  </vbox>
</window>
```

Obrázek 3.8: Ukázka overlay souboru.

Lokalizace - entity a DTD

Je velmi výhodné, pokud máme možnost, snadno změnit veškerý text v aplikaci z jednoho jazyku na druhý. V XUL k tomu slouží tzv. entity (angl. *entities*), které se deklarují v DTD (*Document Type Definition*) souborech. Vše objasní následující příklad 3.9. Můžeme si tedy vytvořit několik `.dtd` souborů podle jazykových verzí a mezi nimi libovolně přepínat podle potřeby.

```
<!-- soubor cestina.dtd -->
<!ENTITY hlavniOkno.nadpis "Pokus">
<!ENTITY tlacitko.odeslat "Odeslat">
<!ENTITY tlacitko.smazat "Smazat">

<!-- soubor main_cesky.xul -->
<?xml version="1.0"?>
<!DOCTYPE window SYSTEM "cestina.dtd">
<window id="hlavniokno-window"
        title="&hlavniOkno.nadpis;"
        xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.
           only.xul">
  <hbox>
    <button id="odeslat-tlacitko" label="&tlacitko.odeslat;">
    <button id="zrusit-tlacitko" label="&tlacitko.zrusit;">
  </hbox>
</window>
```

Obrázek 3.9: Ukázka, jak lze vytvořit několik jazykových verzí aplikace.

Klávesové zkratky

XUL nabízí způsob, jak můžeme definovat klávesové zkratky pomocí atributu `accesskey`. Požadovanou klávesu můžeme definovat buď přímo u daného prvku,

```
<button id="tlacitko1" label="Klikni" accesskey="k"/>
```

nebo prostřednictvím elementu `<key>`. Názorný příklad je uveden na obrázku 3.10. [38]

Commands

Pomocí elementu `<command>` můžeme definovat příkazy, které se mají vykonat. Není nutné jej používat. Příkaz lze samozřejmě napsat i ke konkrétnímu prvku, například tlačítku, ale takto můžeme oddělit uživatelské rozhraní aplikace od obslužného kódu. Vše lépe vysvětlí následující příklad 3.11. [38]

```
<keyset>
  <key id="klikni-key" modifier="alt"
       key="K" oncommand="alert('Klik!')"/>
  <key id="napoveda-key" keycode="VK_F1"
       oncommand="zobrazNapovedu();"/>
</keyset>

<button id="klikni-button" label="Klikni"
        key="klikni-key" command="alert('Klik!');"/>
<button id="napoveda-button" label="Nápověda"
        key="napoveda-key" oncommand="zobrazNapovedu();"/>
```

Obrázek 3.10: Příklad klávesových zkratek.

```
<commandset>
  <command id="hw-cmd" oncommand="alert('Hello World!');"/>
</commandset>

<!-- klasický způsob -->
<button id="hw-button1"
        label="Klikni" oncommand="alert(Hello World!);"/>
<!-- nový způsob -->
<button id="hw-button2"
        label="Klikni" command="hw-cmd"/>
```

Obrázek 3.11: Ukázka použití příkazů `<command>`.

Broadcaster a Observer

Broadcaster a observer jsou mechanismy sloužící ke „sledování stavu" jiných elementů. Broadcaster uchovává nějakou informaci a observer podle ní mění svůj stav. Tyto mechamismy používáme, když chceme měnit stav u více prvků současně, jako je tomu v příkladu na obrázku 3.12. Pomocí změny atributu elementu `<broadcaster>` změníme tento atribut u všech observerů. [33]

```
<broadcasterset>
  <broadcaster id="nepristupne" disabled="true"/>
  <broadcaster id="barva" style="color: red;"/>
</broadcasterset>

<button id="odeslat-tlacitko" label="Odeslat"
        oncommand="odeslatFormular();" observes="nepristupne"/>
<menuitem id="menuitem-ulozit" label="Uložit"
          value="ulozit" observes="nepristupne"/>
<label observes="barva">Nějaký text, momentálně červený...</label>
```

Obrázek 3.12: Ukázka použití příkazů `<command>`.

3.2.2 XUL a CSS

Použití kaskádových stylů v XUL je velmi podobné způsobu použití v HTML. Máme dvě možnosti aplikace stylu. První možností je tzv. *inline styl*, kdy zapisujeme definice vzhledu do `.xul` souboru. Slouží k tomu element `style`:

```
<button class="cerveneTlacitko" style="color: red;" label="Klikni"/>
```

Druhou možností je vložit odkaz na `.css` soubor:

```
<?xml-stylesheet href="externiStyl.css" type="text/css"?>
...
<button class="cerveneTlacitko" label="Klikni"/>
```

Definice vzhledu se pak zapisují do externího souboru:

```
.cerveneTlacitko {
  color: red;
}
```

Praktičtější je druhá varianta, tedy zápis do externího souboru `.css`, který vložíme do `.xul` souboru. Zvýší se tím přehlednost XUL i CSS kódu a budeme mít k dispozici více souborů se stylem, mezi kterými bude možno jednoduše přepínat. [33]

Práce s kaskádovými styly je tedy, až na pár výjimek, stejná jako v HTML. Mozilla si vytvořila několik vlastních rozšíření, která nejsou součástí CSS standardu. Jedná se o speciální příkazy začínající vždy prefixem `-moz-`, například `-moz-opacity` nebo `-moz-binding`. [33]

Všechny elementy v XUL již mají definovaný vlastní výchozí styl, který je umístěný v adresáři `chrome://global/skin/`. Pravděpodobně si s ním ve většině případů vystačíme. Pokud ne, vyvoříme si styl vlastní. Bylo by však náročné tvořit nový vzhled pro všechny prvky, které budeme používat. Proto stačí definovat vzhled jen těm prvkům, u kterých chceme ten původní změnit. Zbytek zdědíme ze stylu výchozího. Stačí tedy vložit více `.css` souborů, kde se jeden z nich bude odkazovat na výchozí vzhled. Pro úsporu místa můžeme využít zápisu, kdy do `.xul` souboru vkládáme pouze jeden styl, ve kterém je odkaz na styl výchozí:

```
@import url(chrome://global/skin/);
#hl-menu {
  ...
}
```

Zobrazovací jádro Gecko podporuje CSS verze 1 i 2 (částečně také verzi 3). [11]

3.2.3 XUL a DOM

V Mozille jsou kromě standardních DOM událostí definovaných W3C standardem implementovány rozšiřující události speciálně pro jazyk XUL (např. `DOMMouseScroll` a další[3]). [5, 42]

Podobně existují také identifikátory vlastností (angl. *properties*) XUL elementů, ke kterým lze přistupovat pomocí JavaScriptu, a metody, prostřednictvím kterých lze k elementům přistupovat, ale které nejsou součástí standardního DOMu a jsou určeny pouze pro jazyk XUL. [34]

Na závěr můžeme zmínit, že Gecko podporuje plně DOM 0 i DOM 1, většinu z DOM 2 a na podpoře DOM 3 se pracuje. [7]

3.2.4 XUL a JavaScript

V každém uživatelském rozhraní je potřeba zajistit obsluhu událostí, jinak bychom měli pouze „nic nedělající" statické rozhraní. V XUL je obsluhy událostí dosaženo pomocí jazyka JavaScript. Opět, podobně jako u kaskádových stylů, máme několik možností, jak vložit JavaScript do XUL souboru. Začneme zápisem javascriptového kódu přímo do `.xul` souboru pomocí elementu `script`:

```
<script type="application/x-javascript">
  function mojeFunkce() {
    // tělo funkce
  }
</script>
```

Druhá možnost zápisu opět využívá element `script`, nyní však s atributem `src` odkazujícím na externí `.js` soubor:

```
<script type="application/x-javascript" src="javascript.js">
```

Třetí možnost je zapsat JavaScriptový kód přímo k danému elementu, jehož obsluhu chceme definovat. Je to ale nepraktické už z důvodu malého prostoru k zápisu, jak můžeme vidět na následujícím příkladu:

```
<window onload="alert('Okno bylo načteno!');"/>
```

[3] Speciální události definované a implementované Mozillou se velmi těžko hledají. Podařilo se mi najít pouze jeden zdroj [5].

Platí zde stejná pravidla jako v kapitole 3.2.2. Je vhodnější, pokud k tomu nemáme speciální důvod, použít druhou variantu, tedy zvlášť `.xul` soubor a zvlášť `.js` soubor. [33]

Obsluhu událostí lze zařídit i jiným zůsobem, a to zavoláním metody `addEventListener` příslušného elementu:

```
<button id="tlacitko1" label="Klikni sem ..."/>

<script type="application/x-javascript">
  function klik(event) {
    alert('Kliknuto!');
  }

  var tlacitko = document.getElementById("tlacitko1");
  tlacitko.addEventListener('command', klik, true);
</script>
```

JavaScript lze ovšem použít i jinak než jen k obsluze událostí, avšak právě ta je hojně využívána.

3.3 Nástroje pro podporu programování

Následující kapitola si klade za cíl krátce představit některé nástroje, které byly použity při tvorbě aplikace internetového obchodu a které mohou vývoj usnadnit či podpořit. Nemilou skutečností je, že žádné specializované vývojové prostředí, jako je například NetBeans pro jazyk Java, pro XUL neexistuje. Nicméně existují nástroje, které nám dokáží práci částečně usnadnit.

3.3.1 Eclipse

Pro vývojové prostředí Eclipse existuje zásuvný modul XULBooster přidávající mimo jiné například automatické doplňování kódu. Dále je užitečné doinstalovat zásuvný modul Aptana obsahující podporu pro generování nápovědy podobné jako je JavaDoc. [48]

Odkazy ke stažení:

- Eclipse - http://www.eclipse.org;
- XULBooster - bohužel původní stránka modulu http://cms.xulbooster.org již není v provozu. Lze jej však dohledat a stáhnout i jinde http://sourceforge.net/projects/xulbooster/;

- Aptana - http://www.aptana.org/.

3.3.2 Firebug a Web Developer

Tyto doplňky pro Firefox se hodí nejenom při práci s jazykem XUL, ale také pro vývoj webových aplikací obsahujicích kaskádové styly, JavaScript, AJAX. Lze je najít na následujících adresách:

- Firebug - http://getfirebug.com/,
- Web Developer - http://chrispederick.com/work/web-developer/.

3.3.3 XULExplorer

Jednoduchý „pomocník" při návrhu uživatelského rozhraní. Je vhodný spíše k naučení základních principů a seznámení se s prvky GUI, než k pokročilému návrhu rozhraní. Internetové stránky aplikace:

- XULExplorer - https://developer.mozilla.org/en/XUL_Explorer.

3.4 Srovnání s dalšími rámci

Zde bude stručně představeno několik rámců pro vývoj webových aplikací.

3.4.1 Microsoft Silverlight

Je platforma určená pro vývoj webových RIA[4] (*Rich Internet application*) aplikací. Pro spuštění aplikace je potřeba mít v prohlížeči nainstalovaný plug-in. Silverlight je podporován na platformách Windows a Mac OS. Existuje i podpora pro Linux s názvem Moonlight[5]. Výhodou Silverlightu je podpora videostreamingu a přehrávání videa ve vysoké kvalitě.

Silverlight je postaven na technologii WPF (*Windows Presentation Foundation*). Vzhled uživatelského rozhraní je definován pomocí XAML (*Extensible Application Markup Language*). Pro programování aplikační logiky lze využít jazyky ze skupiny .NET. Pomocí Silverlightu lze tvořit jak webové, tak desktopové a mobilní aplikace pro plaformu Windows Phone a Symbian. [39]

[4] Webová aplikace podobající se svými vlastnostmi desktopové aplikaci. [20]

[5] Jedná se o OpenSource produkt společnosti Novell, nikoliv Microsoft. V současné době existuje verze Moonlight 4 podporující Silverlight 3 a částečně 4 [17].

XAML

XAML je deklarativní značkovací jazyk vycházející z XML. Zápis obsahuje názvy elementů uvnitř tagů, obdobně jako je tomu například u XUL. Stejně tak jako XUL obsahuje spoustu různých prvků pro definici vzhledu a layoutu. V XAML máme možnost tvořit také jednoduché grafické prvky, úsečku, elipsu, obdelník a další. Existuje zde pochopitelně také možnost vizuální úpravy stylu. Jedná se o technologii podobnou CSS. Další možností je využít tzv. šablon (template), pomocí kterých lze měnit vzhled stávajících komponent. Nová komponenta se bude chovat stejně jako ta původní, jen bude mít jiný vzhled. Pokročilou vlastností je tzv. „data binding". Ten nám umožňuje vytvořit odkaz na proměnnou, jejíž název je při změně aktualizován v celém dokumentu. Další možností jazyka XAML, kterou zde zmíníme, je možnost vytvářet animace. XAML obsahuje také speciální elementy umožňující vkládání mediálních souborů (obrázky, zvuk, video). Z krátkého představení jazyka je vidět, že je zde obsažena velká podpora grafických prvků a médií. [41]

```
<Window x:Class="PureXAML.HelloWorld"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Title="Hello world okno."
  Height="200" Width="300" WindowStartupLocation="CenterScreen">

  <Button x:Name="hwTlacitko" Width="133" Height="24"
          Content="Ahoj světe!" Click="hw_klik"/>

  <x:Code>
    <![CDATA[
      private void btnHelloWorld_Clicked(object sender,
          RoutedEventArgs e)
      {
        MessageBox.Show("Hello World!");
      }
    ]]>
  </x:Code>
</Window>
```

Obrázek 3.13: Ukázka XAML syntaxe.

3.4.2 Adobe Flex

Další technologií pro vývoj RIA aplikací je Adobe Flex. Jedná se o technologii podobnou jako Silverlight od Microsoftu. Aplikace jsou zde kompilovány do `.swf` souborů. Pro spuštění je nutné mít nainstalovaný stejný zásuvný modul jako pro flash, tedy Adobe Flash

Player, který je dostupný pro všechny běžné operační systémy a prohlížeče. Flex nabízí jako alternativu k webovým aplikacím možnost vytvářet také aplikace desktopové. Pro jejich spuštění slouží pak běhové prostředí Adobe AIR.

Pro definici uživatelského rozhraní se používá jazyk MXML, o aplikační logiku se stará skriptovací jazyk ActionScript. Tvorba aplikací bude snazší pro uživatele, kteří mají zkušenosti s HTML. Jde o podobný postup, jen MXML nahrazuje HTML, ActionScript Javascript a CSS styly zůstávají beze změny.

MXML

Je další deklarativní značkovací jazyk z rodiny XML. Opět máme na výběr množství prvků pro definici vzhledu i layoutu, podobně jako v jazyce XAML, stejně tak máme možnost využít data binding. [44]

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml">
  <mx:Script>
    <![CDATA[
      public function hw_klik():void {
        Alert.show('Hello World!');
      }
    ]]>
  </mx:Script>
  <mx:Style>
    Button {
      fontStyle: italic;
    }
  </mx:Style>
  <mx:Panel title="Hello world tlačítko."
            height="75%" width="75%" color="0xffffff">
    <mx:Button id="hwTlacitko" label="Ahoj světe!"
               click="hw_klik();"/>
  </mx:Panel>
</mx:Application>
```

Obrázek 3.14: Ukázka MXML syntaxe.

3.4.3 JavaFX

Poslední zde zmíněnou technologií je JavaFX od Sun Microsystems. Opět jde o platformu pro RIA aplikace a také je pro jejich běh nutný zásuvný modul. Stejně jako u Flexu můžeme tvořit WWW i desktopové aplikace. V případě WWW prostředí se aplikace vkládá do do-

kumentu buď pomocí odkazu v JNLP souboru nebo jako aplet. JavFX pokrývá navíc také mobilní platformu. Pro tvorbu aplikační logiky slouží deklarativní jazyk JavaFX Script, speciálně navržený pro tuto platformu. Narozdíl od předchozích jazyků není JavaFX Script založen na XML, ale zavádí svůj vlastní formát zápisu. Vývojáři mají také možnost importovat Java knihovny, například knihovny grafiky, efektů, multimédií, animací a další. Podporován je také data binding. [44]

```
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.text.Text;
import javafx.scene.text.Font;
Stage {
    title: "Ahoj světe!"
    width: 250
    height: 80
    scene: Scene {
        content: Text {
            font : Font {
                size : 24
            }
            x: 10, y: 30
            content: "Hello World!"
        }
    }
}
```

Obrázek 3.15: Ukázka JavaFX syntaxe.

3.4.4 Shrnutí

Každý z výše uvedených rámců má své klady a zápory. Silverlight je vhodný pro bohaté multimediální aplikace s pokročilými grafickými efekty. Jeho silnou stránkou je streamování audiovizuálního obsahu ve vysoké kvalitě. Stojí za ním Microsoft, podporuje .NET. Adobe Flex se bude spíše hodit pro vysoce interaktivní dynamické stránky. Lze jej bez problémů vložit do klasického HTML. Vyžaduje běhové prostředí Adobe Flash, které je rozšířenější než Silverlight plug-in. JavaFX nemá za sebou zdaleka takovou historii jako předchozí dva zástupci RIA technologií. Vychází z Javy a aplikace lze snadno přenést na mobilní platformu, čímž se dostává před konkurenci. [47]

Podíváme-li se blíže na ukázky kódů na obrázcích 3.13, 3.14, 3.15 a srovnáme-li je s XUL zápisem na obrázku 3.16, zjistíme, že první dva (XAML a MXML) se XUL velmi podobají.

Všechny tři jazyky mají základ v XML, všechny jsou určeny k definici GUI. JavaFX Script se svou syntaxí odlišuje. Mezi výhody XUL (oproti ostatním) bychom mohli zařadit:

- podpora mnoha standardů (HTML, CSS, DOM, JavaScript, RDF, ...);
- podpora dalších technologií, jako je XBL, XPCOM, ...;
- nezávislost na operačním systému (přenositelnost všude tam, kde běží Mozilla).

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet href="chrome://global/skin" type="text/css"?>
<window title="Hello World!"
        xmlns:html="http://www.w3.org/1999/xhtml"
        xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.
           only.xul">
  <script type="application/x-javascript">
    <![CDATA[
      function klikni_hw() {
        alert("Ahoj světe!");
      }
    ]]>
  </script>
  <vbox>
    <buton id="hwTlacitko" label="Klinkni..."
           style="color: red;" onclick="klikni_hw();"/>
  </vbox>
</window>
```

Obrázek 3.16: Ukázka XUL syntaxe.

Kapitola 4

Návrh aplikace

V kapitole 4 přecházíme od teoretické části práce k části praktické. Na následujících stranách je popsán návrh a implementace demonstrační aplikace internetového obchodu. Při návrhu e-shopu vycházíme především ze zkušeností nabytých nakupováním na internetu. Všechny diagramy v této kapitole jsou vytvořeny pomocí softwaru Visual Paradigm for UML[1].

4.1 Popis aplikace

Samotný internetový obchod se skládá ze dvou částí, z části pro běžného uživatele, tedy nakupujícího, a z části pro správce, tedy prodejce. První část je implementována pomocí XHTML, druhá pak pomocí XUL. Pro definici vzhledu jsou použity kaskádové styly CSS. Aplikační logika je v obou částech řešena pomocí JavaScriptu, na serveru pracuje PHP. Pro ukládání dat volíme databázový systém MySQL kvůli jeho podpoře v PHP, rozšíření a také kvůli podpoře transakcí.

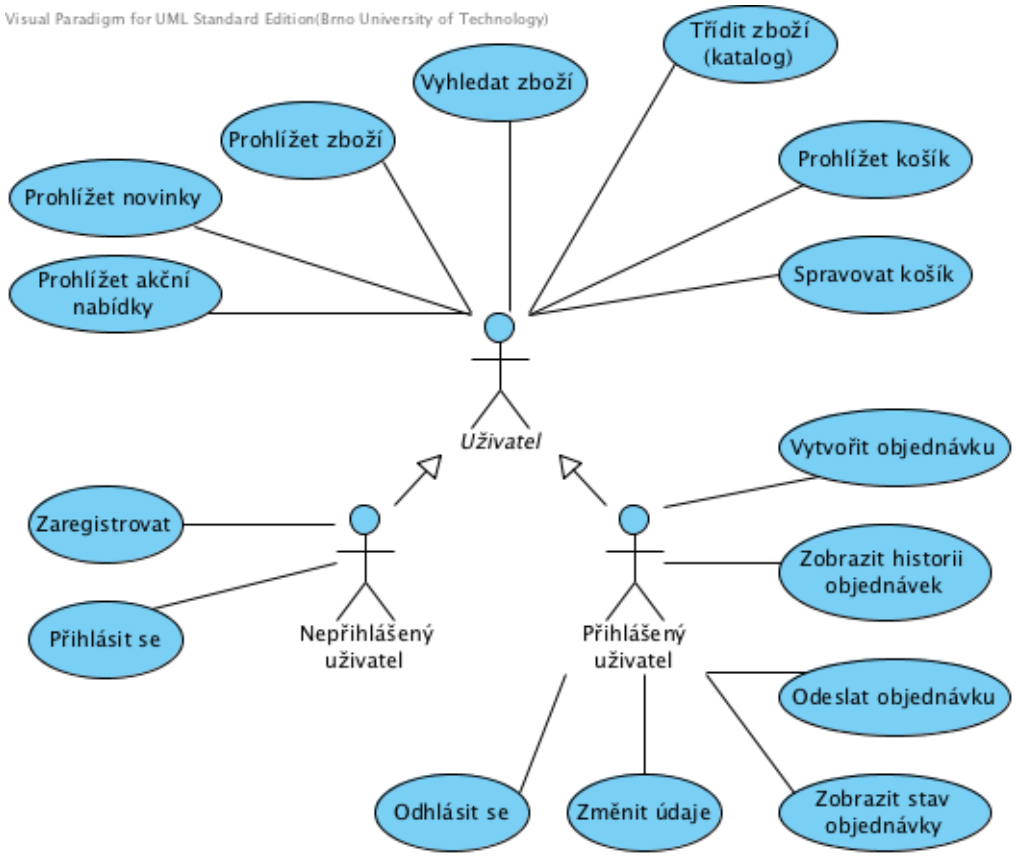
Nakupující může vykonávat běžné činnosti, mezi které patří například prohlížení zboží, vyhledávání, přidávání do košíku a další aktivity, které běžné e-shopy poskytují. Správce (prodejce) má možnost vyřizovat objednávky, objednávat zboží do skladu a podobně. Přehled všech případů užití je znázorněn na obrázcích 4.1 až 4.3 a popsán v kapitole 4.2.

Nedílnou součástí internetového obchodu je databáze, kde jsou uloženy mimo jiné informace o nabízených produktech, skladových zásobách, registrovaných uživatelích (více v kapitole 4.3).

[1] http://www.visual-paradigm.com/

4.2 Use case diagram

Diagram případů užití pro nakupujícího uživatele (obrázek 4.1) obsahuje dvě role, *přihlášený uživatel* a *nepřihlášený uživatel*. Protože většina případů užití je pro oba aktéry společná, je vhodné vytvořit třetí, abstraktní, roli *uživatel*.

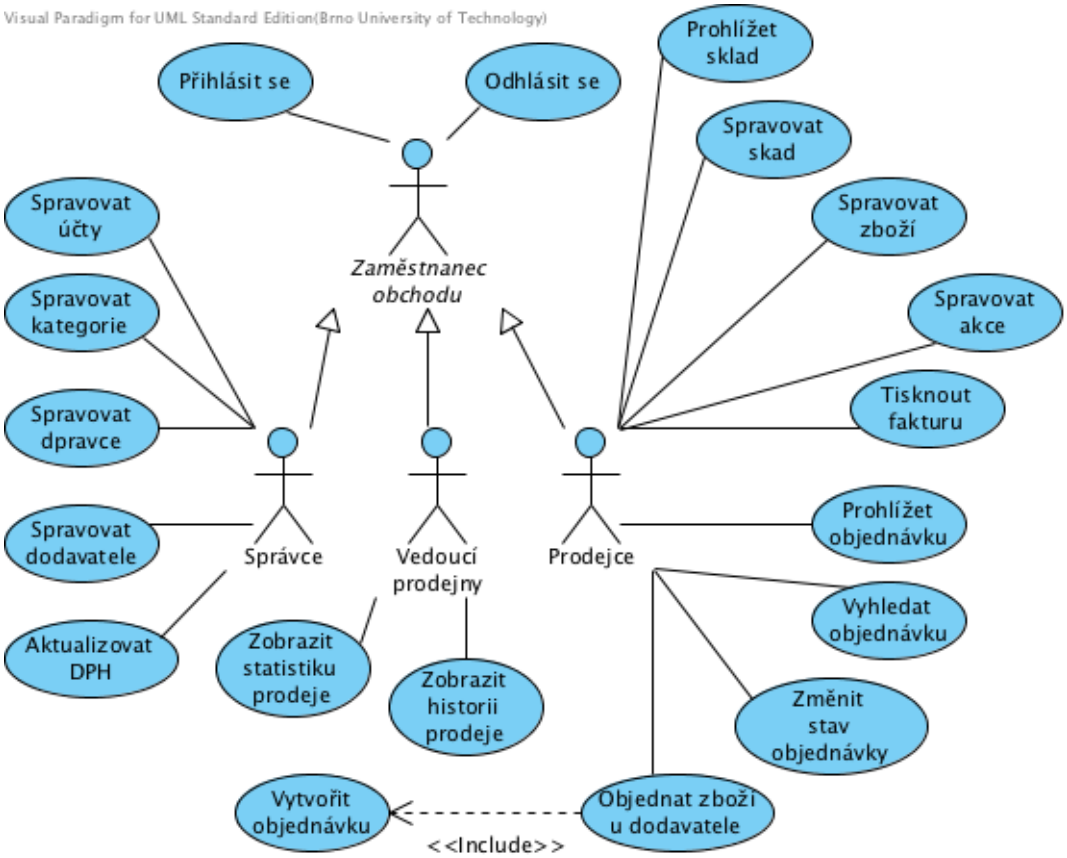


Obrázek 4.1: Use case diagram uživatel.

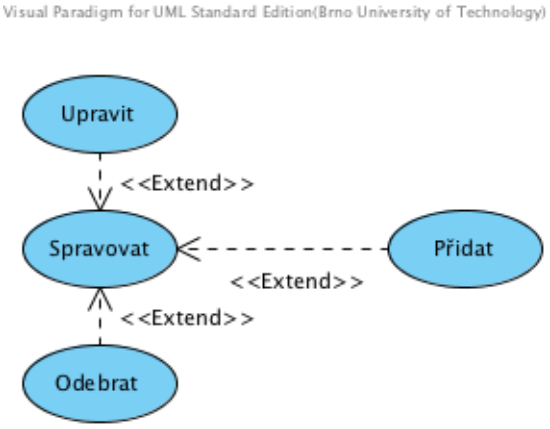
V diagramu části pro zaměstnance (obrázek 4.2) jsou role tři, *správce*, *prodejce* a *vedoucí prodejny*. Opět vytvoříme abstraktní roli *zaměstnanec obchodu* pro společné případy užití. Vedoucí prodejny a prodejce jsou vzájemně velmi podobné role, s tím rozdílem, že vedoucí je nadřízený. Má tedy vyšší pravomoce (může nahlížet do statistik prodeje a podobně), může však také vykonávat stejné úkony jako jeho podřízení (například vyřídit objednávku), ale není to jeho hlavní pracovní náplň.

Na závěr je ještě nutné zmínit poslední use case diagram, kterým je 4.3, vytvořený čistě jen pro přehlednost předchozích use case diagramů, které obsahují několik případů užití začínajících slovem *spravovat*. Každá správa znamená možnost *přidávat*, *odebírat* a *upravovat*

dané položky, například zboží v nákupním košíku, uživatelské účty apod. Protože se tato správa často opakuje, je pro ni vytvořen zvláštní use case diagram.



Obrázek 4.2: Use case diagram zaměstnanec.

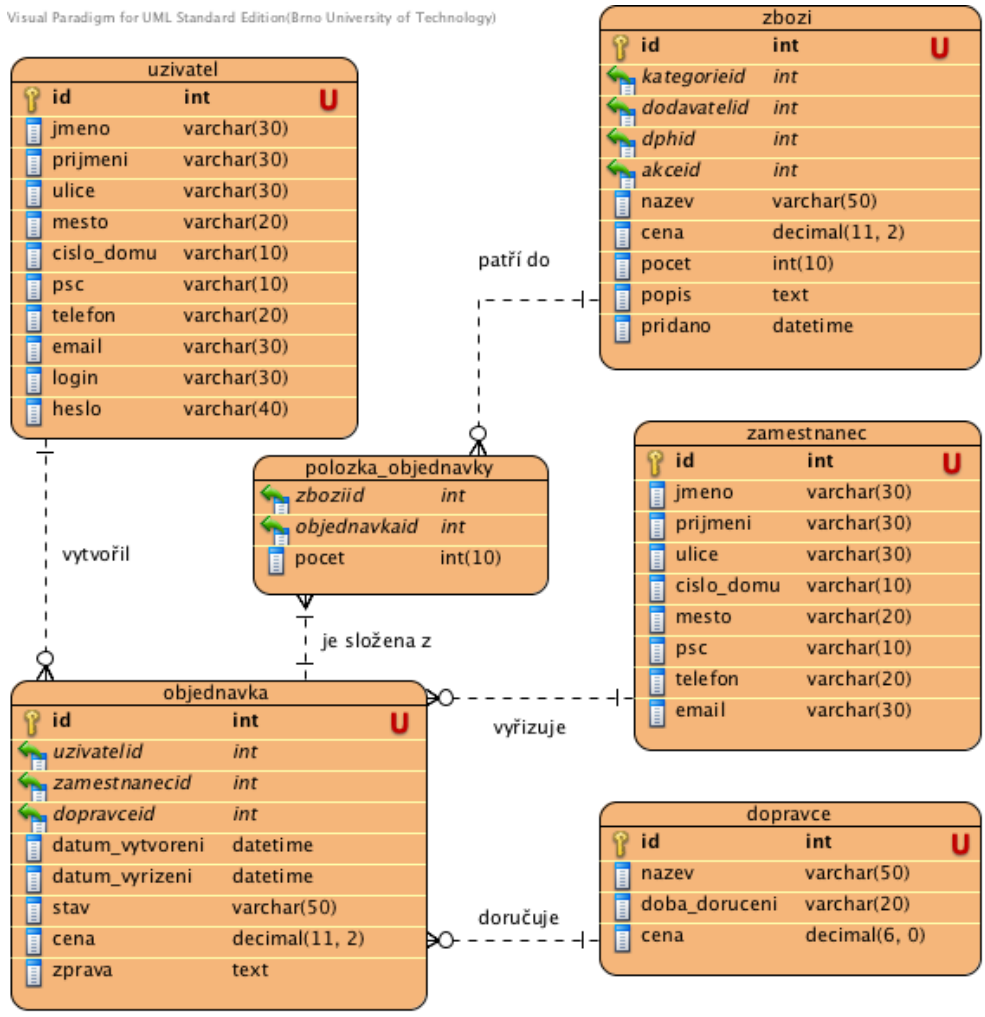


Obrázek 4.3: Use case diagram spravovat.

4.3 ER diagram

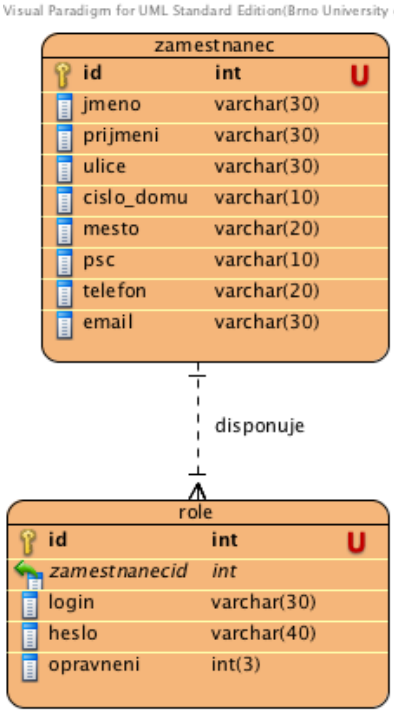
S návrhem databáze, která slouží jako úložiště dat, je spojena tvorba ER diagramu. Tento diagram znázorňuje, jaké údaje jsou do databáze ukládany a vztahy mezi nimi. Podobu databáze elektronického obchodu můžeme vidět na obrázcích 4.4 až 4.7.

První část diagramu (obrázek 4.4) znázorňuje uživatele a tvorbu objednávky. Uživatel vytváří objednávky skládající se z jednotlivých položek zboží. Tyto objednávky doručuje dopravce a vyřizuje zaměstnanec. Atribut *pridano* je v entitě *zbozi* proto, že se podle něj určuje nově přidané zboží za určité období.



Obrázek 4.4: ER diagram, část pro uživatele a objednávky.

Na druhé části diagramu (obrázek 4.5) vidíme role zaměstnance v systému. Každý zaměstnanec může v systému vystupovat pod několika různými rolemi s odlišnými pravomocemi. Pro každou roli v systému bude mít jiné přihlašovací jméno.

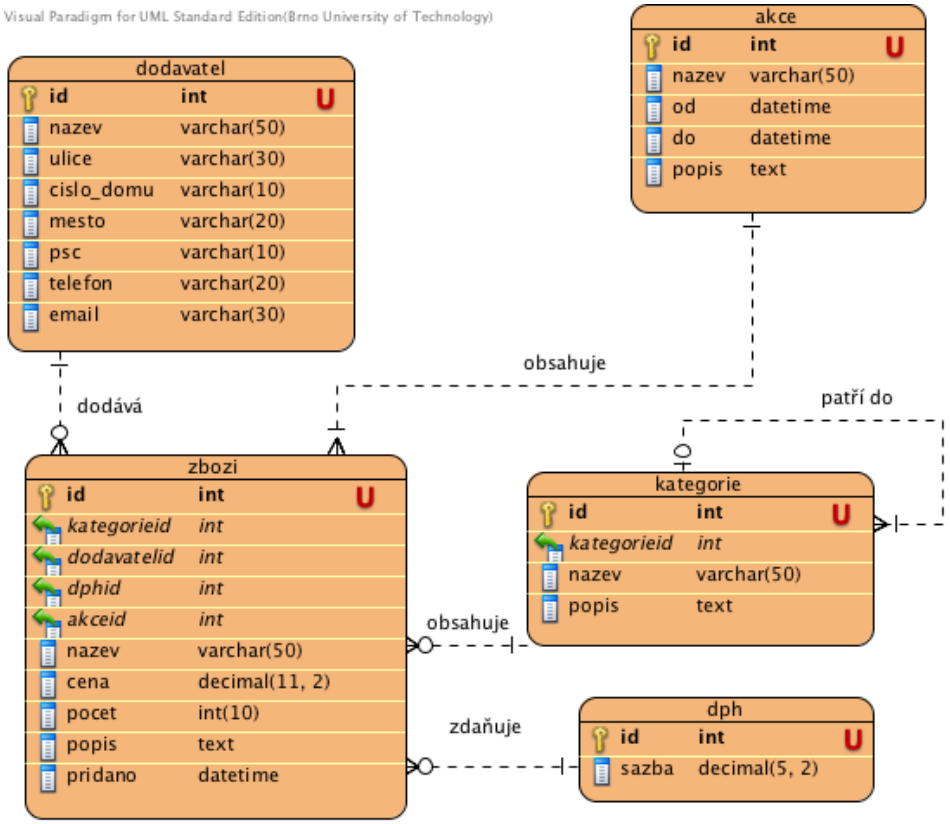


Obrázek 4.5: ER diagram, část pro zaměstnance.

Hlavní entitou na třetí části diagramu (obrázek 4.6) je *zbozi*, patřící do určité kategorie, popřípadě také akce, dodávané dodavatelem a zdaněné sazbou DPH. Každá akce má stanovenou dobu platnosti. Kategorie může obsahovat podkategorie a ty mohou opět obsahovat podkategorie atd.

Pokud nějaké zboží není skladem, má zaměstnanec možnost ho objednat u dodavatele (čtvrtá část diagramu, obrázek 4.7). Dodávka se stejně jako objednávka skládá z jednotlivých položek.

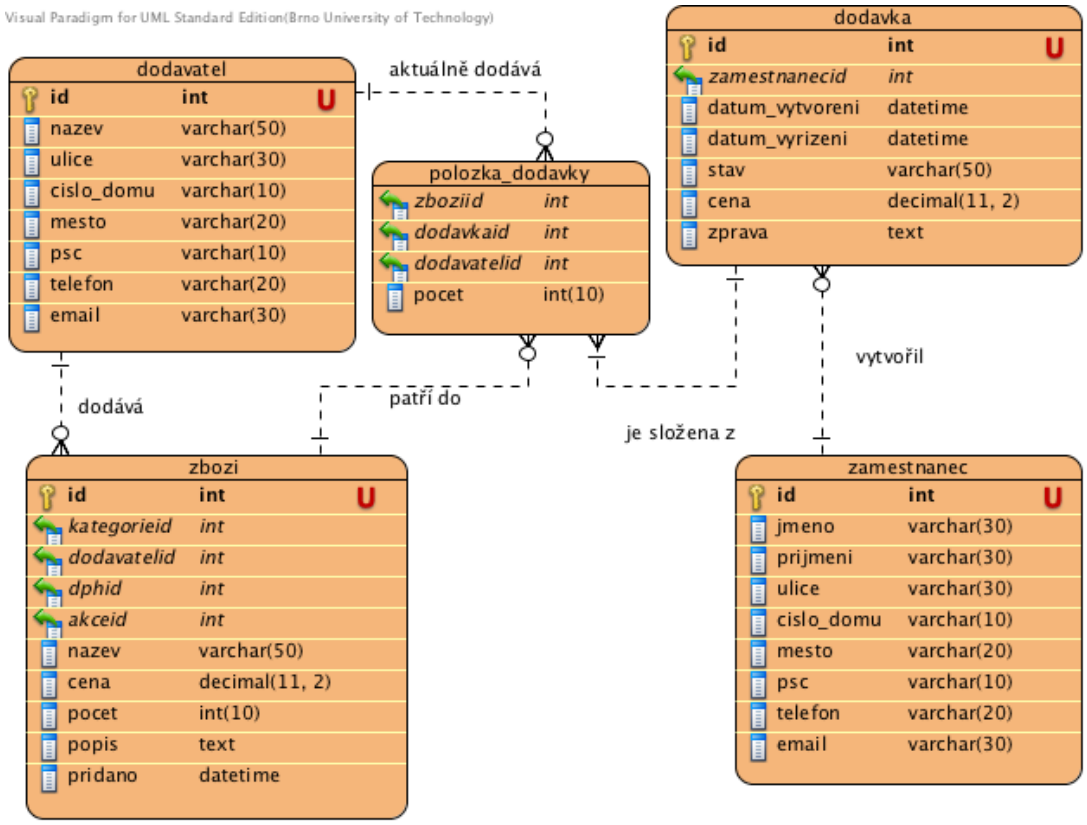
Všechna hesla se ukládají ve skryté podobě, tzv. „zahashované". Před samotnou transformací hesla se k němu navíc přidává náhodný řetězec. Ten je pokaždé jiný a určuje se podle loginu. Bez tohoto řetězce by dvě stejná hesla byla uložena v databázi pod stejným otiskem, což nechceme.



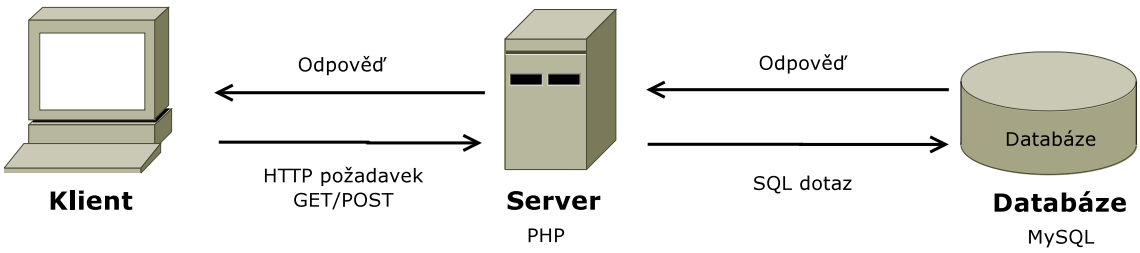
Obrázek 4.6: ER diagram, část pro dodávku.

4.4 Komunikace

S daty uloženými na serveru je potřeba nějakým způsobem pracovat. Komunikaci klienta s databázovým serverem zajišťuje PHP. Klient vytvoří požadavek na data. Odešle jej na webový server pomocí protokolu HTTP, který požadavek zpracuje, vytvoří SQL dotaz pro databázový server a odešle jej. Databázový server ho zpracuje, získá data, o která si klient požádal a pošle je zpět na webový server. Ten data zpracuje do požadované formy a předá klientovi. Schéma komunikace je znázorněno na obrázku 4.8.



Obrázek 4.7: ER diagram, část 4.



Obrázek 4.8: Ukázka komunikace.

Kapitola 5

Implementace

Následující kapitola se zaměřuje na praktickou část práce. Popisujeme zde obě části internetového obchodu.

5.1 Databáze

Jak již bylo řečeno, zvolená databáze je MySQL. Při tvorbě tabulek vycházíme z poznatků popsaných v 4.3. Celkem je potřeba vytvořit čtrnáct tabulek.

Pro ukázkovou aplikaci je použita verze MySQL 5.1.56 na fakultním serveru eva.

5.2 Část pro nakupující

V této sekci se zaměříme na první část aplikace sloužící uživatelům internetového obchodu, kteří v něm budou nakupovat.

5.2.1 Popis aplikace

Grafické uživatelské rozhraní

Je tvořeno pomocí XHTML a CSS.

Struktura

Aplikace se skládá z několika souborů. Jedná se o sobory `.php`, `.css` a `.js`. Chceme-li mít v `.html` souboru kromě běžných HTML tagů také PHP skripty, musíme ho uložit

s koncovkou `.php`. Tím dáme najevo serveru, že je v něm obsažen také PHP kód. Proto nemáme žádné `.html` soubory.

Opakující se části, jako je menu, záhlaví a zápatí, funkce pro připojení k databázi, funkce pro výpis kategorií a další, jsou umístěny v samostatných souborech a vkládány pomocí PHP příkazu `include`, respektive `require` a `require_once`.

JavaScriptové funkce jsou umístěny v několika souborech ve složce `script`.

Uchovávání dočasných údajů

Údaje o přihlášení jsou uloženy pomocí PHP seancí. Ukládáme *id uživatele*, *login*, *jméno*, *příjmení* a *čas přihlášení*. Čas přihlášení používáme kvůli neaktivitě, která je nastavena na třicet minut. Pokud uživatel nevykoná žádný pohyb, bude po této době automaticky odhlášen. Vzhledem k tomu, že seance jsou serverová záležitost, narozdíl od například cookies, není v aplikaci další způsob ověřování těchto údajů.

Pro nákupní košík nejsou seance vhodné, zavřením okna prohlížeče se smažou, narozdíl od cookies, které přetrvávají, dokud nejsou smazány uživatelem nebo nevyprší jejich platnost. Položky v nákupním košíku ukládáme ve formátu „*id - počet kusů - cena jednoho kusu,*":

```
var platnost = new Date();
platnost.setDate(platnost.getDate() + 365); // + 365 dnů
var polozka = id+"-"+pocet+"-"+cena+",";
document.cookie="Kosik="+polozka+";expires="+platnost.toGMTString()+";";
```

Pro práci s nákupním košíkem a cookies slouží JavaScriptové funkce v souborech `kosik.js` a `susenky.js`.

Knihovny

V aplikaci je použita JavaScriptová knihovna (soubory `jquery.min.js`, `jquery-ui.min.js`) jQuery[1] pro vytvoření dialogového přihlašovacího okna. Knihovna podléhá MIT a GPL licenci.

[1] http://jquery.com

5.2.2 Odesílání dat

Probíhá pomocí formulářů `<form>`. V atributu `action` je určen PHP skript, který požadavek zpracuje. Způsob předávání dat je určen atributem `method`. Zde máme na výběr metody `POST` a `GET`:

```
<form action="zpracuj.php" method="get">
<form action="zpracuj.php" method="post">
```

V případě metody `POST` jsou data odesílána na server skrytě, narozdíl od metody `GET`, která předává data jako součást URL hlavičky. Můžeme je tedy vidět v řádku adresy. Jednotlivé položky se oddělují znakem `&`:

```
stranka.php?polozka1=hodnota1&polozka2=hodnota2
```

V `.php` souboru získáváme odeslané hodnoty pomocí superglobálních proměnných `$_GET` a `$_POST`:

```
$prom_get = $_GET["polozka"];
$prom_post = $_POST["polozka"];
```

Pomocí formulářových prvků získáváme data, odesíláme je na server, tam jsou zpracována pomocí PHP a odeslána zpět v podobě XHTML.

5.2.3 Obslužné PHP skripty

Vykonávají určitou činnost, pro kterou jsou navrženy, například ukládají data do databáze. Systém odesílání známe z předchozí podkapitoly 5.2.2. Podle hodnoty získané proměnné se rozhoduje, jaká akce bude vykonána.

5.3 Část pro prodejce

Nyní si popíšeme druhou část aplikace, správcovskou část pro prodejce.

5.3.1 Popis aplikace

Grafické uživatelské rozhraní

Je tvořeno pomocí XUL a CSS. Při definování vzhledu jednotlivých elementů se můžeme dostat do situace, kdy chceme stávající (výchozí) styl nahradit stylem vlastním. Ve většině

případů stačí určit vlastní vzhled, popřípadě ještě přidat klauzuli `!important` k dané vlastnosti. Mohou nastat případy, kdy ani jedna z těchto možností nebude fungovat. Děje se tak pravděpodobně z důvodu, že námi definovaná CSS vlastnost je přepsána nějaku jinou z defaultního vzhledu. V tom případě je vhodné otevřít si výchozí `.css` soubor a zjistit, která vlastnost to způsobuje, a tu potom nahradit.

Struktura

Jádro aplikace tvoří soubor `index.xul`, do kterého jsou podle potřeby vkládány další části. Hlavní menu, lišta s ikonami a stavový řádek jsou opakující se prvky, proto jsou uloženy v samostatných souborech a vkládány jako tzv. overlays. Některá okna, jako například okno s nápovědou nebo informacemi o aplikaci, se otevírají samostatně. Pomocí elementu `iframe` jsou do souboru `index.xul` vkládány potřebné soubory. Děje se tak pomocí změny atributu `src`, kterou zajišťuje funkce `zmenStranku(soubor)`:

```
zmenStranku("objednavky.xul");
```

Vše uvnitř rámce `iframe` je samostatný dokument. Chceme-li přistupovat z nadřazeného dokumentu k prvkům uvnitř rámce, musíme specifikovat, že přistupujeme k elementu umístěném v daném rámci. Analogický postup platí pro přístup z rámce k prvkům v nadřazeném dokumentu:

```
// přístup z nadřazeného doumentu do rámce
// nula určuje přístup do prvního rámce
window.frames[0].document.getElementById("prvek_uvnitr");

// přístup z rámce do nadřazeného dokumentu
window.parent.document.getElementById("prvek_vne");
```

Až na pár výjimek platí, že ke každé tabulce z databáze náleží minimálně jeden `.xul` soubor, jeden `.js` soubor a jeden `.php` soubor. Například:

- `objednavky.xul` - obsahuje uživatelské rozhraní,
- `objednavky.js` - obsahuje obslužné funkce,
- `objednavky.php` - obsahuje kód pro zpracování dat a komunikaci s databází.

Všechny JavaScriptové soubory jsou umístěny v adresáři `script`. Další `.php` soubory jsou dle potřeby vkládány pomocí příkazu `require_once`. Takto se vkládá skript pro připojení k databázi, pro výpis kategorií, pro hashování hesla atd. Důležitým souborem je také `ajax.js`, obsahující kód pro práci s technologií AJAX.

Nápověda je vytvořena pomocí RDF a prostého HTML. Soubor `napoveda.rdf` obsahuje strukturu nápovědy a v souboru `napoveda.html` je umístěn vlastní text nápovědy. Oba dva zpracovává soubor `napoveda.xul`, tvořící výsledné okno nápovědy.

V aplikaci se často setkáváme se vstupním polem (`textbox`), do kterého se zadávají údaje různých datových typů. Jména ukládáme v databázi jako datový typ `varchar`, datum a čas jako `datetime`, pěněžní částky jako `decimal` apod. Každý datový typ, ale také každý atribut dané entity, je vhodné před uložením do databáze ověřit. Protože se zadávané údaje často opakují, například adresa se vyskytuje hned u třech entit, jsou pro usnadnění práce vytvořeny pomocí `xbl` nové prvky, včetně validačních funkcí. Každý prvek obsahuje několik položek:

- `label` s názvem,
- `textbox` pro vstupní údaje,
- `label` označující povinnou/nepovinnou položku,
- `label` pro hlášení o výsledku validace,
- vše je uvnitř elementu `hbox` kvůli zarovnání.

Při změně hodnoty se vyvolá obsluha události `onChange`, zavolá se validační funkce a je zobrazen výsledek kontroly. Všechny nové elementy jsou umístěny v souboru `xbl.xml` ve složce `xbl`:

```
<box class="email" nazev="E-mail:"
     maxlength="30" size="30" povinne="*"/>
```

Předchozí výpis ukazuje, jak lze jednoduše vložit prvek pro zadávání a kontrolu e-mailové adresy. Klasickým způsobem by byl zápis o poznání delší:

```
<hbox align="center">
  <label value="E-mail:" style="width: 10em;"/>
  <textbox onchange="validace();" maxLength="30" size="30"/>
  <label value="*"/>
  <label/> <!-- pro výpis případné chyby -->
</hbox>
```

Uchovávání dočasných údajů

Protože remote XUL aplikace nepodporují práci s cookies pomocí `document.cookie` (více viz bug 144795 v [28]), ukládají se dočasné údaje o přihlášení pomocí globálního objektu `sessionStorage`, který je dostupný po celou dobu, kdy máme okno prohlížeče otevřené. Otevřením nového okna nebo nové záložky vzniká další objekt. Pro náš účel, uložení informace o aktuálním přihlášení, tento objekt plně postačuje. Pracujeme s ním následovně:

```
// uložení
sessionStorage.setItem("login", "xraulo00");

// vyzvednutí
sessionStorage.getItem("login");

// odebrání
sessionStorage.removeItem("login");
```

Při přihlášení ukládáme *jméno* a *příjmení zaměstnance*, *oprávnění*, *id zaměstnance*, *id* jeho *role* a klíč *session_id*.

Knihovny

Pro tvorbu PDF při tisku faktury se používá PHP třída FPDF[2]. Jde o knihovnu uvolněnou pod permissive license. Dále je použita služba Google Chart Tools[3] pro tvorbu grafů u statistiky a historie prodeje.

5.3.2 Odesílání dat

Probíhá pomocí technologie AJAX. Nejprve získáme data, která se budou odesílat. K tomu nám slouží převážně metoda `getElelemtById`. Tímto způsobem získáme všechny potřebné hodnoty:

```
var promenna = document.getElementById("id_objednavky").value;
```

Nyní je potřeba vytvořit instanci objektu `XMLHttpRequest`. Před samotným odesláním musíme specifikovat parametry volání serveru. Mezi povinné parametry metody patří určení metody volání (GET nebo POST) a URL. V aplikaci internetového obchodu je vytvořena funkce `pozadavekXhr(url,vyberFce)`. Parametr `vyberFce` určuje, která funkce

[2] http://www.fpdf.org/
[3] http://code.google.com/apis/chart/

bude zpracovávat odpověď při úspěšném požadavku. Naše funkce zajistí vytvoření objektu, určení parametrů a odeslání:

```
pozadavekXHR("zpracuj.php?id=1&jmeno=Karel&prijmeni=Vomacka",
             "tvorba-uctu-zamestnance");
```

Při úspěšně provedené akci vrací server stavový kód 200. Změna stavu je uložena v atributu `readyState`. Atribut sledujeme a při každé jeho změně se volá obslužná funkce, která zjistí stav a podle toho se vykoná další akce. V našem případě jde o funkci `zpracujXhr()`. Při neúšpěšném požadavku se zobrazí chybové hlášení, při úspěšném se zpracuje odpověď. Funkce `zpracujXhr()` podle proměnné `vyberFce` vykoná příslušnou obsluhu odpovědi.

5.3.3 Obslužné PHP skripty

Při odesílání požadavku na PHP skript posíláme s daty také řetězec identifikující požadovanou akci, kterou má skript vykonat. Pomocí tohoto řetězce se rozhodujeme, jak budeme dále postupovat. Jak již bylo řečeno, jeden `.php` soubor je určen pro obsluhu několika požadavků od často stejnojmenného `.xul` souboru. Tedy například při práci s objednávkami potřebujeme mít možnost načíst objednávky z databáze, měnit jejich stav, prohlížet detail každé objednávky. Každá z těchto činností vyžaduje jiný dotaz do databáze a jiné zpracování. Proto je potřeba vybírat mezi těmito různými akcemi, jak je znázorněno na následujícím příkladu:

```
if($_GET["akce"] == "objednavky-seznam") nactiSeznamObjednavek();
else if($_GET["akce"] == "objednavka-detail") nactiDetailObjednavky();
...
```

Po určení akce provedeme požadované příkazy, nejčastěji jde o získání dat z databáze. Načtená data poté upravíme do podoby vhodné pro odeslání. Odesíláme ve dvou podobách, první je XML a druhá JSON. Obecně platí, že pro větší objem položek používáme XML a pro menší JSON. Protože při použití PHP funkce `json_encode()` se vyskytly problémy se špatným kódováním českých znaků, byla vytvořena nová PHP třída `mujJson` vracející data v JSON formátu. Ukázky, jak mohou vypadat výstupy, jsou znázorněny na obrázcích 5.1 a 5.2. První obrázek 5.1 je odpověď serveru na dotaz požadující seznam všech dopravců. Na druhém obrázku 5.2 žádáme o detail dopravce.

```
<DOPRAVCI>
  <DOPRAVCE>
    <ID>1</ID><NAZEV>PPL</NAZEV>
  </DOPRAVCE>
  <DOPRAVCE>
    <ID>2</ID><NAZEV>Česká pošta</NAZEV>
  </DOPRAVCE>
  <DOPRAVCE>
    <ID>3</ID><NAZEV>Osobní převzetí</NAZEV>
  </DOPRAVCE>
</DOPRAVCI>
```

Obrázek 5.1: Příklad dat ve formátu XML.

```
{"dop_id": "1","dop_naz": "PPL","dop_cen": "125","dop_dob": "2"}
```

Obrázek 5.2: Příklad dat ve formátu JSON.

Kapitola 6

Závěr

Cílem práce bylo seznámit se z jazykem XUL, s postupem vývoje webových aplikací pomocí tohoto jazyka, dále srovnání s dalšími rámci pro vývoj webových aplikací a demonstrace použití XUL na aplikaci internetového obchodu.

Mezi výhody XUL patří rozsáhlá podpora stávajících standardů a jazyků, jako CSS, XML, DOM, XBL apod. Množství zabudovaných komponent posiluje jazyk XUL, především ve tvorbě uživatelského rozhraní. Syntax není složitá. Jak jsme mohli vidět na příkladech v podkapitole 3.4, je zde podoba s dalšími značkovacími jazyky pro tvorbu uživatelského rozhraní.

XUL má ovšem i své nevýhody. Neexistuje příliš mnoho zdrojů, odkud by se daly čerpat informace. Vyjma internetových stránek MDN[1] a několika málo knih se informace hledají s obtížemi. Často je tak programátor nucen navštívit různá diskuzní fóra a čerpat z nich. XUL také nepodporují ostatní prohlížeče. Tento fakt omezuje okruh jeho použití. Nicméně pro správcovská rozhraní různých internetových aplikací, jako například internetových obchodů nebo redakčních systémů, kde můžeme „nařídit" uživatelům použití Firefoxu, je vhodný.

Ve srovnání s (X)HTML má XUL rozsáhlejší seznam komponent, které nám usnadňují práci. Přidáme-li XBL s možností tvorby znovupoužitelných komponent a RDF, získáme tak kvalitní nástroj pro tvorbu uživatelského rozhraní. V XUL máme navíc možnost použití libovolného prvku z (X)HTML.

Při nasazení vytvořené aplikace do reálného provozu by bylo vhodné více se zaměřit na zabezpečení obou jejích částí, zejména na kontrolu zadávaných dat, která se ukládají

[1] Mozilla Developer Network, http://developer.mozilla.org/.

do databáze. Dále pak případně přizpůsobit některé její části požadavkům konkrétního zákazníka.

6.1 Možná rozšíření

Jedno z možných rozšíření může být vytvoření aplikace jako desktopové, spoušťěné pomocí XULRunneru, komunikující se vzdáleným serverem. V tomto případě je nutné použít technologii či přístup umožňující cross-domain AJAX požadavky. Takto vytvořená desktopová aplikace není omezována jako vzdálené XUL aplikace. Může například přistupovat k místním souborům a složkám na disku. Dalším problémem vzdáleného XUL jsou některé chyby (angl. *bugs*). Jedna z těchto chyb je zmíněná v podkapitole 5.3, kde jsme narazili na problém s `document.cookie`. Mezi další nedostatky patří mimo jiné také špatná funkce elementu `wizzard` či problém s externím DTD souborem. Ostatní lze najít v [28]. Dále od verze 4 přestává Firefox standardně vzdálené XUL podporovat. Lze jej však pomocí doplňku povolit. Postup je nastíněn v příloze A.

Literatura

[1] *Alphabetical list of all XUL elements* [online]. Poslední modifikace: 16. dubna 2010. [cit. 2011-04-16].
URL https://developer.mozilla.org/en/XUL_Reference

[2] *Cascading Style Sheets* [online]. Poslední modifikace: 22. února 2011. [cit. 2011-02-22].
URL http://www.w3.org/Style/CSS/

[3] *Cascading Style Sheets* [online]. Poslední modifikace: 16. února 2011. [cit. 2011-02-22], Wikipedie.
URL http://en.wikipedia.org/wiki/Cascading_Style_Sheets

[4] *DOM CURRENT STATUS* [online]. [cit. 2011-02-22].
URL http://www.w3.org/Standards/techs/dom#w3c_all

[5] *DOM events* [online]. Poslední modifikace: 1. března 2011. [cit. 2011-03-01].
URL http://en.wikipedia.org/wiki/DOM_events

[6] *Dynamic HTML* [online]. Poslední modifikace: 19. února 2011. [cit. 2011-02-20], Wikipedie.
URL http://en.wikipedia.org/wiki/Dynamic_HTML

[7] *Gecko FAQ* [online]. Poslední modifikace: 16. prosince 2010. [cit. 2011-03-01].
URL https://developer.mozilla.org/en/gecko_faq

[8] *Gecko* [online]. Poslední modifikace: 9. února 2011. [cit. 2011-02-20], Wikipedie.
URL http://cs.wikipedia.org/wiki/Gecko

[9] *Getting started with XULRunner* [online]. Poslední modifikace: 9. března 2011. [cit. 2011-04-13].
URL https://developer.mozilla.org/en/Getting_started_with_XULRunner

[10] *Introduction to XBL* [online]. Poslední modifikace: 9. února 2011. [cit. 2011-02-25].
URL https://developer.mozilla.org/en/XUL_Tutorial/Introduction_to_XBL

[11] *Mozilla CSS support chart* [online]. Poslední modifikace: 18. prosince 2010. [cit. 2011-03-01].
URL https://developer.mozilla.org/en/Mozilla_CSS_support_chart

[12] *MySQL* [online]. Poslední modifikace: 15. února 2011. [cit. 2011-02-23].
URL http://en.wikipedia.org/wiki/MySQL

[13] *MySQL: The world's most popular open source database* [online]. [cit. 2011-02-23].
URL http://www.mysql.com/

[14] *PHP: Hypertext Preprocessor* [online]. Poslední modifikace: 23. února 2011. [cit. 2011-02-23].
URL http://www.php.net/

[15] *RDF: Často kladené dotazy* [online]. Poslední modifikace: 30. listopadu 2003. [cit. 2011-04-18].
URL http://dsic.zapisky.info/archive/RDF-FAQ-20031130/

[16] *RDF/XML Syntax Specification (Revised)* [online]. [cit. 2011-04-18].
URL http://www.w3.org/TR/REC-rdf-syntax/

[17] *Release Notes Moonlight4 Preview - Mono* [online]. [cit. 2011-04-14].
URL http://www.mono-project.com/Release_Notes_Moonlight4_Preview

[18] *Resource Description Framework* [online]. Poslední modifikace: 9. dubna 2011. [cit. 2011-04-18].
URL http://en.wikipedia.org/wiki/Resource_Description_Framework

[19] *Resource Description Framework* [online]. [cit. 2011-04-18].
URL http://www.xul.fr/en-xml-rdf.html

[20] *Rich Internet application* [online]. Poslední modifikace: 21. března 2011. [cit. 2011-04-14].
URL http://en.wikipedia.org/wiki/Rich_Internet_application

[21] *Using Remote XUL* [online]. Poslední modifikace: 29. listopadu 2010. [cit. 2011-04-13].
URL https://developer.mozilla.org/en/Using_Remote_XUL

[22] *XBL 1.0 Reference* [online]. Poslední modifikace: 12. října 2011. [cit. 2011-02-25].
URL https://developer.mozilla.org/en/XBL/XBL_1.0_Reference

[23] *XHTML 1.0 The Extensible HyperText Markup Language (Second Edition)* [online]. [cit. 2011-02-21].
URL http://www.w3.org/TR/xhtml1/

[24] *XHTML 1.1 - Module-based XHTML - Second Edition* [online]. [cit. 2011-02-21].
URL http://www.w3.org/TR/xhtml11/

[25] *XML Binding Language (XBL) 2.0* [online]. Poslední modifikace: 12. října 2011. [cit. 2011-02-25].
URL http://www.w3.org/TR/2007/CR-xbl-20070316/

[26] *XMLHttpRequest* [online]. [cit. 2011-02-22].
URL http://www.w3.org/TR/XMLHttpRequest

[27] *XUL* [online]. Poslední modifikace: 27. prosince 2010. [cit. 2011-02-20], Wikipedie.
URL http://en.wikipedia.org/wiki/XUL

[28] *XUL:Remote XUL bugs* [online]. Poslední modifikace: 13. dubna 2007. [cit. 2011-04-26].
URL https://wiki.mozilla.org/XUL:Remote_XUL_bugs

[29] *XULRunner 1.9 Release Notes* [online]. Poslední modifikace: 6. ledna 2010. [cit. 2011-04-13].
URL https://developer.mozilla.org/en/xulrunner_1.9_release_notes

[30] *XULRunner* [online]. Poslední modifikace: 7. února 2011. [cit. 2011-02-20].
URL http://en.wikipedia.org/wiki/XULRunner

[31] *XULRunner tips* [online]. Poslední modifikace: 20. srpna 2010. [cit. 2011-04-13].
URL https://developer.mozilla.org/en/XULRunner_tips#Using_Firefox_3_to_run_XULRunner_applications

[32] Asleson, R.; Schutta, N. T.: *AJAX: vytváříme vysoce interaktivní webové aplikace.* Brno: Computer Press, 2006, 269 s, ISBN 80-251-1285-3.

[33] Bosvell, D.; King, B.; Oeschger, I.; et al.: *Creating Applications with Mozilla.* 1st ed. O'Reilly Media, 2002, 480 p, ISBN 0-596-00052-9.

[34] Bullard, V.; Smith, K. T.; Daconta, M. C.: *Essential XUL Programming.* Wiley Computer Publishing, 2001, 432 p, ISBN 0-471-41580-4.

[35] Burget, R.; Hruška, T.: *Internetové aplikace (WAP) IV.: část Programování serveru (PHP) studijní opora* [online]. Verze únor 2007.
URL https://www.fit.vutbr.cz/study/courses/WAP/private/opory/OporaWAP4ProgramovaniServeru.pdf

[36] Burget, R.; Zeman, D.: *Tvorba webových stránek ITW studijní opora* [online]. Verze 20.10.2006.
URL https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/ITW-IT/texts/opora_itw_061020.pdf

[37] Castagnetto, J.; Rawat, H.; Schumann, S.; et al.: *PHP: programujeme profesionálně.* Praha: Computer Press, 2001, 656 s, ISBN 80-7226-310-2.

[38] Deakin, N.: *XUL Tutorial* [online]. [cit. 2011-04-02].
URL https://developer.mozilla.org/en/XUL_Tutorial

[39] Hejduk, P.: *Silverlight jako platforma nejen pro internetové prezentace.* Bakalářská práce, České vysoké učení technické, Fakulta elektrotechnická, Praha, 2010.

[40] Hruška, T.: *Internetové aplikace (WAP) VI.: část Programování klienta (JavaScript) studijní opora* [online]. Verze únor 2007.
URL https://www.fit.vutbr.cz/study/courses/WAP/private/opory/OporaWAP6ProgramovaniKlienta.pdf

[41] Kolda, J.: *Tvorba internetových aplikací pomocí technologie Microsoft Silverlight.* Bakalářská práce, Jihočeská univerzita, Pedagogická fakulta, České Budějovice, 2009.

[42] McFarlane, N.: *Rapid Application Development with Mozilla.* New Jersey: 1st ed. Prentice Hall, 2003, 800 p, ISBN 0-13-142343-6.

[43] Máčel, L.; Kužela, A.; Hruška, T.: *Internetové aplikace (WAP) V.: část AJAX studijní opora* [online]. Verze únor 2007.
URL https://www.fit.vutbr.cz/study/courses/WAP/private/opory/OporaWAP5Ajax.pdf

[44] Novák, J.: *Analýza RIA metod a technik*. Diplomová práce, Masarykova univerzita, Fakulta informatiky, Brno, 2009.

[45] Schaffer, S. M.: *HTML, XHTML a CSS: bible [pro tvorbu WWW stránek]*. 4. vyd. Praha: Grada, 2009, 647 s, ISBN 978-802-4728-506.

[46] Staníček, P.: *CSS Kaskádové styly: kompletní průvodce*. Brno: Computer Press, 2003, 178 s, ISBN 80-7226-872-4.

[47] Vávra, J.: *Rámce pro vývoj webových aplikací*. Diplomová práce, Masarykova univerzita, Fakulta informatiky, Brno, 2009.

[48] Vídeňský, M.: *Mozilla jako vývojová platforma*. Diplomová práce, Vysoké učení technické, Fakulta informačních technologií, Brno, 2008.

[49] Zakas, N. Z.; Krejčí, L.: *JavaScript pro webové vývojáře: programujeme profesionálně*. Vyd. 1. Brno: Computer Press, 2009, 832 s, ISBN 978-802-5125-090.

Příloha A

Prostředí pro běh aplikací

Jak již bylo řečeno v kapitole 3.1, je pro spuštění aplikace nutné použít XULRunner nebo prohlížeč schopný rendrovat XUL, například dříve zmiňovaný Firefox[1].

Při tvorbě tzv. remote XUL aplikací je nutné otevírat stránku, kde se aplikace nachází, v prohlížeči Firefox. Od verze 4 však přestává Firefox podporovat vzdálené XUL a XBL1, mimo jiné kvůli bezpečnostním problémům. Existuje však doplněk Remote XUL Manager[2], pomocí kterého jsme schopni remote XUL domény povolit. Dále musíme zajistit, aby náš webový server rozeznal XUL soubor a dokázal ho správně zpracovat. U webového serveru Apache stačí přidat následující řádek do souboru `mime.types`:

```
application/vnd.mozilla.xul+xml .xul
```

Alternativním řešením je přidat do souboru `httpd.conf` podobný řádek, jako v předchozím případě:

```
AddType application/vnd.mozilla.xul+xml .xul
```

Popřípadě můžeme přidat stejný řádek do souboru `htaccess` v adresáři, kde máme `.xul` soubory. Pokud používáme PHP, stačí použít funkci `header()`:

```
<?php
  header('Content-type: application/vnd.mozilla.xul+xml');
?>
<!-- XUL -->
```

[1] Dalším prohlížečem by pak mohl být SeaMonkey. Dále bude v textu zmiňován pro stručnost pouze Firefox.

[2] https://addons.mozilla.org/en-US/firefox/addon/remote-xul-manager/

Webový server však není nutný, dokument lze v prohlížeči otevřít pomocí `Soubor -> Otevřít soubor` nebo `Soubor -> Otevřít adresu`. [21]

Při tvorbě desktopových aplikací, tedy aplikací, které poběží lokálně na našem počítači, je nutné mít nainstalovaný XULRunner nebo Firefox. Postup instalace a nastavení XULRunneru se liší podle operačního systému.

1. Stažení

Stažení XULRunneru pro daný operační systém, například z https://developer.mozilla.org/en/XULRunner.

2. Instalace

Ve Windows rozbalíme archiv, například do `C:\xulrunner`, poté zaregistrujeme příkazem spuštěným v příkazovém řádku:

```
xulrunner.exe --register-global //pro všechny uživatele
xulrunner.exe --register-user   //pro aktuálního uživatele
```

V Mac OS otevřeme `.pkg` soubor.

V Linuxu podobně jako ve Windows:

```
xulrunner --register-user          //pro všechny uživatele
sudo ./xulrunner --register-global //pro aktuálního uživatele
```

3. Spuštění aplikace

Ve Windows z adresáře spouštěné aplikace vykonáním příkazu (pokud máme `xulrunner.exe` v systémové proměnné `PATH`):

```
xulrunner.exe application.ini
```

V Mac OS existuje několik variant, nejjednodušší, dle mého názoru, je následující:

```
/Library/Frameworks/XUL.framework/xulrunner-bin
    "/<full path>/application.ini"
```

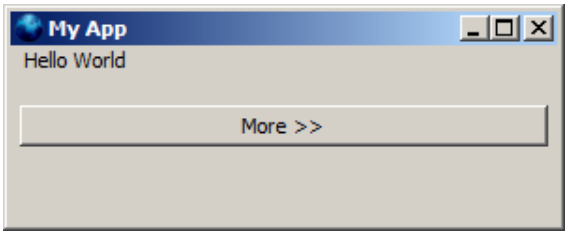
V Linuxu z adresáře aplikace příkazem:

```
xulrunner application.ini
```

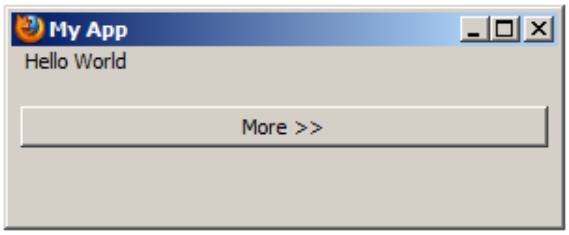
Detailní postup nastavení, popis spouštění aplikací a další informace, které zde nebyly zmíněny, nebo byly zmíněny jen stručně, lze nalézt v [9, 29]. Pokud z nějakého důvodu nechceme nebo nemůžeme použít XULRunner, nabízí se alternativní řešení, kterým je Firefox. Spuštění je pak následující:

- Windows - `firefox.exe -app path\to\application.ini`,
- Mac OS - `/Applications/Firefox.app/Contents/MacOS/firefox-bin -app /path/to/application.ini`,
- Linux - `firefox -app path/to/application.ini`.

Na obrázcích A.1 a A.2 jsou vidět okna jednoduchých aplikací. První aplikace byla spuštěna



Obrázek A.1: Okno jednoduché aplikace otevřené pomocí běhového prostředí XULRunner.

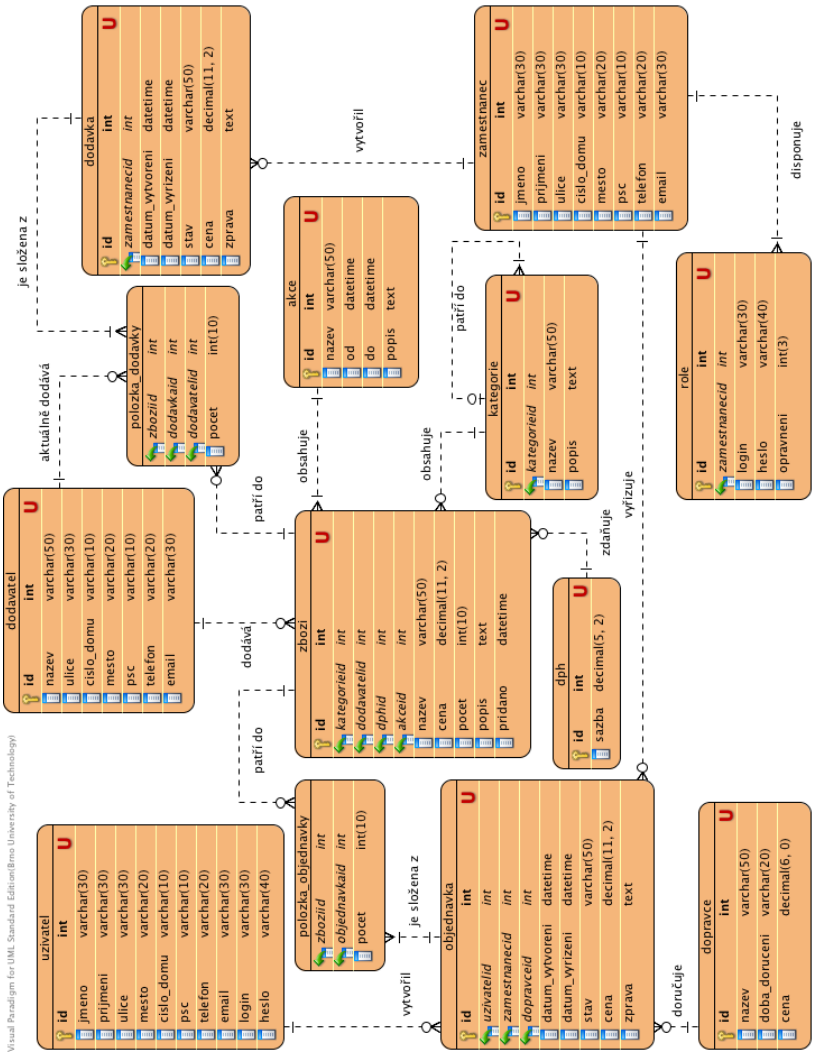


Obrázek A.2: Okno jednoduché aplikace otevřené pomocí prohlížeče Firefox 3.

pomocí XULRunneru, druhá alternativním způsobem, tedy pomocí Firefoxu, konkrétně verze 3. Jak je vidět z obou obrázků, okna jsou téměř totožná. [9, 31]

Příloha B

ER diagram



Obrázek B.1: Kompletní ER diagram.

Příloha C

Obsah přiloženého CD

Přiložené CD obsahuje:

- adresář `src` se zdrojovými kódy aplikace,
- soubor `readme.txt` se základním popisem aplikace a jejich částí,
- adresář `text` s textem práce,
- adresář `obrazky` se snímky obrazovek aplikace,
- adresář `prevzato` obsahující převzaté knihovny a obrázky,
- adresář `db` s databázovými skripty.

Příloha D

Přihlašovací údaje

Internetový obchod je dočasně umístěn na fakultním serveru na adresách:

- http://www.stud.fit.vutbr.cz/~xraulo00/bp/nakupujici/index.php,
- http://www.stud.fit.vutbr.cz/~xraulo00/bp/prodejce/index.xul.

První adresa odkazuje na část pro nakupující. Přihlašovací údaje jsou `uzivatel/heslo` a `uzivatel2/heslo2`. Na druhé adrese je umístěna část pro zaměstnance. Přihlašovací údaje jsou `prodejce/heslo`, `vedouci/heslo` a `administrator/heslo`.